Layered Security: Gradient Boosting Meets Naive Bayes for Intrusion Detection

A
Project Report

Submitted in partial fulfilment of the
Requirements for the award of the Degree of

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

By

Piyush Bhuyan	**1602-21-737-014**
Vootla Krishna Sai Srinivas	**1602-21-737-026**

Under the guidance of

Mr. K. Srinivasa Chakravarthy

Assistant Professor

Department of Information Technology

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University)

Ibrahimbagh, Hyderabad-31

2025

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University)

Hyderabad-500 031

Department of Information Technology

DECLARATION BY THE CANDIDATE

We, **Piyush Bhuyan** and **Vootla Krishna Sai Srinivas** bearing hall ticket numbers, **1602-21-737-014** and **1602-21-737-026** hereby declare that the project report entitled **Layered Security: Gradient Boosting Meets Naive Bayes for Intrusion Detection** under the guidance of **Mr. K. Srinivasa Chakravarthy, Assistant Professor**, Department of Information Technology, Vasavi College of Engineering, Hyderabad, is submitted in partial fulfilment of the requirement for the award of the degree of **Bachelor of Engineering** in **Information Technology.**

This is a record of the bonafide work carried out by us and the results embodied in this project report have not been submitted in part or in full to any other university or institute for the award of any other degree or diploma.

Piyush Bhuyan
1602-21-737-014

Vootla Krishna Sai Srinivas
1602-21-737-026

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University)

Hyderabad-500 031

Department of Information Technology

BONAFIDE CERTIFICATE

This is to certify that the project entitled **Layered Security: Gradient Boosting Meets Naive Bayes for Intrusion Detection Title** being submitted by **Piyush Bhuyan** and **Vootla Krishna Sai Srinivas** bearing **1602-21-737-014** and **1602-21-737-026** in partial fulfilment of the requirements for the award of the degree of Bachelor of Engineering in Information Technology is a record of the bonafide work carried out by them under my guidance.

Mr. K. Srinivasa Chakravarthy
Assistant Professor
Internal Guide

Dr. K. Ram Mohan Rao
Professor & HOD, IT

External Examiner

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of this project would not have been possible without the kind support and help of many individuals. We would like to extend our sincere thanks to all of them.

We wish to convey our special thanks to **Dr. S. V. Ramana, Principal** of **Vasavi College of Engineering and Management** for providing facilities.

We are very much thankful to **Dr. K. Ram Mohan Rao, Professor & HOD, Department of Information Technology,** for his kind support and for providing necessary facilities to carry out the work.

It is with immense pleasure that we would like to take the opportunity to express our humble gratitude to **Mr. K. Srinivasa Chakravarthy, Assistant Professor, Department of Information Technology** under whom we executed this project. We are also grateful to our project coordinator **Mrs. C. Sireesha, Assistant Professor, Department of Information Technology** for her guidance. Their constant guidance and willingness to share their vast knowledge made us understand this project and its manifestations in great depths and helped us to complete the assigned tasks.

Not to forget, we thank all other faculty and non-teaching staff, who had directly or indirectly helped and supported us in completing our project in time.

ABSTRACT

The flood of devices connected to the Internet has significantly increased the chances of advanced cyberattacks, and as a result, the IDS are required to evolve to improve the accuracy, efficiency, and scalability. The IDS can largely be classified into two approaches: one which is signature-based and another which is anomaly-based. Both approaches have inherent limitations in that they are unable to define unknown or advance attacks, such as Remote-to-Local (R2L) and User-to-Root (U2R) attacks.

The above limitations, therefore, justify the hybrid machine learning models as alternative approaches, one of which is the Double-Layered Hybrid Approach (DLHA). DLHA, which started out using Naive Bayes (NB) to categorize frequent attacks and Support Vector Machines (SVM) for rare attacks, has proven beneficial in the detection of intrusion. However, the inefficiency of SVM when dealing with real-world applications is shown in its poor performance in handling imbalanced datasets. The study proposes an upgrade of the second layer of the Deep Learning-based Hybrid Approach (DLHA) by replacing the SVM by Gradient Boosting algorithms such as XGBoost and LightGBM.

These algorithms would prove to be exceptionally valuable with high detection accuracy because of their iterative learning methodology, fast training due to the histogram-based learning employed by LightGBM, improved handling of imbalanced data through built-in mechanisms like the parameter scale_pos_weight in XGBoost, and the other general advantages of these algorithms to present great potential to provide feature importance insights for optimizing complex IDS frameworks.

Table of Contents

List of Figures iii

List of Tables iv

List of Abbreviations v

1 INTRODUCTION 1

1.1 Problem Statement – Overview 1

1.2 Motivation 2

1.3 Scope & Objectives of the Proposed Work 2

1.4 Organization of the Report 3

2 LITERATURE SURVEY 6

3 PROPOSED SYSTEM 15

3.1 Methodology 15

3.1.1 Architecture Diagram 18

3.1.2 Functional Modules 19

4 EXPERIMENTAL SETUP & RESULTS 22

4.1 System Specifications 22

4.1.1 Software Requirements 23

4.1.2 Hardware Requirements 24

4.2 Description 25

4.3 Results & Test Analysis 32

5 CONCLUSION AND FUTURE SCOPE 36

REFERENCES 38

APPENDIX 40

Main Code

Plagiarism Report last page

Research Paper

LIST OF FIGURES

Fig. No	Description	Page No
3.1.	Architecture Diagram	18
4.1.	Description of column in KDDTrain+	26
4.2.	Graph comparing Performance Metrics between two DLHA model	33

LIST OF TABLES

Table No.	Table Name	Page No
2.1.	Literature Table	9
4.1.	Distribution of Classes in Original KDDTrain+	31
4.2.	Distribution of Classes in Enhanced KDDTrain+	31
4.3.	Overall Model Performance on KDDTest+	32
4.4.	Detection Rate per Attack Type	33

LIST OF ABBREVIATIONS

Abbreviation	Full Form
DLHA	Double Layered Hybrid Approach
DoS	Denial of Service
FAR	False Alarm Rate
GAN	Generative Adversarial Network
ICFS	Intersectional Correlated Feature Selection
IDS	Intrusion Detection System
NB	Naive Bayes
R2L	Remote to Local
ROC	Receiver Operating Characteristic Curve
SVM	Support Vector Machine
U2R	User to Root
XGBoost	eXtreme Gradient Boosting

1. INTRODUCTION

1.1. Problem Statement – Overview

The growing proliferation of internet-connected devices has contributed in no small measure to increased network vulnerabilities, exposing systems to ever more sophisticated cyberattacks. One example of cyber-attacks is Denial of Service (DoS), Probe, Remote-to-Local (R2L), and User-to-Root (U2R) attacks, which IDS can, in one way or another, prevent from occurring. The common technique of using signatures in traditional IDS does a good job in detecting known threats; however, zero-day attacks or unidentified patterns of intrusion often escape detection.

On the contrary, the anomaly-based method for IDS offers more flexibility to detect novel threats but suffers high returns of false positives and performs poorly in complex scenarios. The hybridization of different machine learning algorithms has emerged as a very promising option to overcome the limits of any single classifier. The Double-Layered Hybrid Approach (DLHA) using Naive Bayes (NB) algorithm for frequent attacks and Support Vector Machines (SVMs) for rare attacks has proven to be very effective in improving detection especially with detection rates for rare types of attacks such as R2L and U2R. However, SVM is computationally expensive, sensitive to hyperparameter tuning, and poor on highly imbalanced datasets, which limit its performance and scalability in real-world applications. In this paper, we replace the SVM in DLHA Second Layer with Gradient Boosting algorithms such as XGBoost and LightGBM to solve a problem.

These algorithms have proven successful in managing class imbalances, providing iterative learning for increased accuracy, and minimizing both training and inference time. Better detection of complex and rare attack types is expected from incorporating advanced methods into the IDS framework while improving overall efficiency and scalability by doing so. This paper proposes replacing the SVM in DLHA second layer with Gradient Boosting algorithms such as XGBoost and LightGBM to address these challenges. First, these algorithms are proven to have a good solution to the problem of class imbalance besides providing iterative learning towards better accuracy while significantly reducing both training and inference

times. With these, the IDS framework can be said to develop better detection capabilities for new and complex attack patterns, as well as increased overall efficiency and scalability.

1.2. Motivation

One of the important and emerging problems for research in cyberspace is how to remain always one step ahead of the increased challenges to the timely detection of computer attacks and what the legacy solutions have not been able to provide about those challenges. Advanced attack methods, like Remote-to-local (R2L) and User-to-root (U2R), mimic very much normal traffic patterns that they are difficult to capture.

Though these attacks occur with lower frequency, they represent one of the most serious threats against critical systems, such as military, financial, and healthcare systems. Moreover, Support Vector Machines (SVM)—widely used in IDS frameworks—are costly in computation and need very fine tuning of hyperparameters, thus more difficult to employ for online applications. It also has a poor revealer on imbalanced datasets where rare attacks are poorly represented.

Promising alternatives to this problem area, Gradient Boosting such as XGBoost and LightGBM, have proven capabilities beyond that of SVM with regard to both accuracy and efficiency. They are also better positioned with regard to the problem of imbalanced data and add, through feature importance, explainability into complex and large-scale intrusion detection tasks. Some studies are really here to take care of these issues by formulating a robust scalable efficient IDS framework towards improving detection rates against rare and complex attacks.

1.3. Scope & Objectives of the Proposed Work

The present work describes the improvement of the performance of the Double-Layered Hybrid Approach (DLHA) for intrusion detection systems by incorporating gradient boosting algorithms such as XGBoost and LightGBM in the second layer in place of SVMs. In addition to this, it is expected that the enhanced detection rates for rare attacks such as R2L and U2R will be achieved while at the

same time computational overhead will be reduced. This is to focus on improving detection accuracy through the iterative property of gradient boosting algorithms and also attempt to reduce false alarm rates by these algorithm types through an established higher capability of better separation between the normal and malicious.

Thus, we have come up with a solution to the imbalanced datasets problem using the scale_pos_weight type of incorporation of XGBoost-and gradient synergies enhancement in detecting overly underrepresented classes. It also measures the time CPU of training and inference of this proposed framework against its predecessor, the original DLHA, before establishing a greater improvement in computational efficiency. Finally, an extensive framework would be validated on variousdatasets under real-world conditions to investigate its scalability and adaptability towards practical deployment. The enhancement of the performance of the Double-Layered Hybrid Approach (DLHA) as an intrusion detection system will be done by stitching Gradient Boosting algorithms such as XGBoost and LightGBM into the second layer in lieu of using SVMs. This is expected to result in higher detection rates for rare attacks such as R2L and U2R while reducing operational overhead.

This study is aimed at improving the detection rate through the iterative nature of gradient boosting algorithms, which have been found to be most effective in rare attack cases. It is also aimed at decreasing the false alarm rates since gradient boosting models would more differentiate between normal and malicious traffic. Addressing the challenge of imbalanced datasets should therefore incorporate such mechanisms into gradient boosting, such as scale_pos_weight in XGBoost, for enhancing detection on underrepresented attack categories. Besides, it also measures the training and inference time taken by the proposed framework compared with the original DLHA so as to show how great a change has been made in the computational efficiency.

Eventually, the refined configuration would be validated on diverse datasets under real-world conditions to assess its scalable dimension and adaptability for practical deployment.

1.4. Organization of the Report

Introduction:

The ever increasing and rapid number of internet-connected devices have thus made increasingly complex cyber-attacks which pose quite a threat to Intrusion Detection Systems. Most present-day methods are often inefficient to address rare or unknown attacks such as R2L and U2R due to class imbalances and computational inadequacies. The present report aims at addressing such conditions by establishing an improved Double-layered hybrid approach (DLHA) replacing SVM with Gradient Boosting algorithms such as XGBoost and LightGBM. The objective under consideration is penning down the mechanisms that improve accuracy while reducing false alarm rates and, at the same time, ensuring scalability in detecting both frequent and rare attack types.

Literature Survey:

Several frameworks of IDS have been developed from the standpoint of modular architectures, feature selection models, and hybrid approaches. A module-based approach is suggested by Mora-Gimeno et al. for holistic detection, but the majority of the present systems suffer poor scalability. The Accuracy of Deep Learning-based IDs was high, but computational expenses were incurred. Sparse autoencoders had a solution in dimensionality reduction but found no old datasets relevant. Most studies show that most single classifiers cannot effectively deal with the problem of class imbalance, thus needing ensemble and hybrid learning for better detection.

Existing System:

It classified the attacks such as DoS and Probe into frequent attacks in Layer 1 using Naive Bayes and other attacks such as R2L and U2R in Layer 2 using SVM. It was also used in the course of this research using the NSL-KDD dataset, which included subsets KDDTrain+ and KDDTest+. The system gained quite a good detection rate of 93.11% with an F1 score of 90.57%, but it had a very high rate of false alarms of 11.82% and severe class imbalance in rare classes, which hinders the adaptability of the system to unseen threats.

Proposed Methodology:

The improved deep learning hybrid architecture utilizes XGBoost and LightGBM in place of SVM in layer 2 for the classification of input space. The architecture uses generative adversarial networks for the augmentation of rare classes, followed by data transformation steps that may include normalization, one-hot encoding, and PCA. Layer 1 Naive Bayes considers only frequented attacks, while Layer 2 XGBoost focuses on attacks found at rare instances. This approach would increase accuracy and lower false negatives, thus enhancing scalability for the real world.

Future Scope:

The forthcoming research will delve into self-adapting methods like reinforcement learning for coping with emerging attack patterns in future attacks. Furthermore, putting efficient deep learning models into practice and achieving dynamic feature selection will be the topics of interest for improving accuracy. Finally, integrating distributed systems for real-time scalability and post-processing to reduce false alarms will be the prime areas of development.

References:

These references include a study on hybrid IDS models, data augmentation based on GAN, algorithms like Gradient Boosting, and limitations of existing systems, which are ideally suited for building up the complete palette of foundation resources for this research.

2. LITERATURE SURVEY

Numerous Intrusion Detection Systems make use of hybrid Machine Learning models for better performance and accuracy.

Mora-Gimeno et al. [1] designed a two-stage modular system architecture for intrusion detection with the aim of improving the performance of detection through diverse techniques. Simultaneous detection is in the first stage, in which multiple intrusion detection methods provide a holistic view of the behavior of applications, thereby raising the probability of detection significantly and the modular structure of the system ensures flexibility and adaptability. However, this multi-detector approach incurs much computational complexity, which makes the system less efficient and not scalable.

Li et al. [2] presented the Gravitational Search Algorithm (GSA), in which particles, being mass objects, are attracted to those with higher fitness values. Particles converge toward the optimal solution without environmental cues by gravitational interactions. GSA has been effectively applied to feature selection, where it helps identify relevant features by using the mass-based attraction mechanism. The drawback is slow convergence with decreasing gravitational force for subsequent iterations, mainly for larger or high-dimensional problem spaces. Further, if overexploited, later iterations will see the particles crowd towards a local optimum prematurely and, in most cases, stop their further search towards the global optimum.

In earlier proposed solutions, a deep learning-based IDS for cyber-attack detection on CANs with 99% accuracy that uses an LSTM autoencoder have been put forward [3]. As a problem, though, one of the main weaknesses with LSTM is that their detection usually takes so long as such networks can't process inputs in parallel.

Other solutions have proposed a sparse autoencoder-based Network Intrusion Detection System (NIDS) [4],[5], reporting that the proposed model achieved 79.1% accuracy for multiclass classification on the NSL-KDD dataset. However, the NSL-KDD dataset is considered outdated and suffers from class imbalance, which limits the generalizability and effectiveness of the model in real-world applications.

To directly use network data as features, features are simply created by applying one-hot encoding to each byte value of the packet data [6]. Therefore, in this case, the total feature size is larger than the packet size.

Siddiqi et al. [7] discussed how high-dimensional data lowers the anomaly detection rate for machine learning-based Network Intrusion Detection Systems (NIDS). The authors have also mentioned the inefficiency of converting non-image network traffic data into image formats for anomaly detection. Moreover, the authors mentioned that over-reliance on large feature sets compromises the real-time detection capabilities of NIDS.

Pajouh et al. [8] proposed a two-layer dimension reduction and two-tier classification (TDTC) model for intrusion detection systems (IDS). The model uses a dimension reduction module to reduce high-dimensional data and a two-tier classification module combining Naive Bayes (NB) and a certainty factor version of k-NN for detecting suspicious behavior. However, the model was tested only on the NSL-KDD dataset, which makes the results questionable in broader applications because generalizability to other datasets is still a concern.

Leevey and Khoshgoftaar [9] studied a survey on intrusion detection models, using the CICIDS2018 dataset, which is comprised of common data and attacks used for NIDS. Some challenges highlighted are overfitting when NIDS uses the CICIDS2018 dataset, under-cited class imbalances within the dataset, and the inefficient process of data cleansing that may have adverse impacts on the IDS models.

Akbani et al. [10] have discussed the failure of machine learning algorithms applied to imbalanced datasets, which arise from the usual Occam's razor principle in the construction of classifiers. As a result of this, these algorithms do not work well on imbalanced datasets. In case of imbalanced data, classifiers default to the majority class as the baseline and classify most of the samples as negative; thus, the performance will be biased and the minority class detection accuracy will also be poor.

Ahmed and Mahmood [11] proposed a semi-supervised learning method for NIDS anomaly detection in network traffic pattern analysis that combines the use of both

supervised and unsupervised learning. They argue that, if there is no labeled data, the unsupervised learning often suffers with low accuracy of detection and high false alarm rates. Their proposed approach aims at reducing dependence on labeled data while enhancing the exploitation of the unlabeled data for high accuracy of detection.

Wisanwanichthan et al. [12] proposed a double-layered hybrid approach for network intrusion detection by NB and SVM[13]. The classification performance delivered 88.97% in accuracy, 90.57% in the F1 score, 88.17% in the precision, and 93.11% in detection with an 11.82% false alarm ratio utilizing the KDDTrain + dataset for training. The approach was further tested to be effective in a relatively much smaller set, KDDTrain+_20Percent, which it also obtained acceptable values, including accuracy of 87.55%, an F1 score of 89.19%, precision at 88.17%, detection rate at 90.24%, and had a 11.83% false alarm rate. The system uses an Intersectional Correlated Feature Selection (ICFS) as its technique in feature selection. This does reduce the dimensionality and increases the efficiency, but its effectiveness is solely dependent on the quality and representativeness of the training data, which might limit the adaptability of the system to unseen or evolving threats.

Idhammad et al. [14] discussed semi-supervised machine learning methodology for DDoS detection, noting the use of the SMOTE oversampling method for resampling minority classes in datasets. The authors used this technology to oversample minorities by 50 to 110 times, thus achieving a balance within the dataset. However, excessive oversampling of minority samples can increase false alarms, which is a challenge to optimal detection performance.

Most AI-based intrusion detection systems (IDS), except for decision forests, rely on base learning models that do not aggregate decisions for efficient IDS management [15], [16]. These models tend to suffer from high false positive rates, meaning large companies receive thousands of security alerts every day [17]. Moreover, they may have high false negative rates, which pose great risks in safety-critical network applications [18]. Previous studies have mainly emphasized on the classification accuracy of single AI algorithms without taking into consideration the strengths of multiple techniques combined. This gap underscores the immediate

need to include ensemble learning methods to improve the efficacy of IDS [19], [20].

Table 2.1. Literature Table

S.No	Authors	Paper	Methodology	Result
1	Mora Gimeno, Francisco José	Intrusion detection system based on integrated system calls graph and neural networks.	Two-Stage Modular Ensemble System	Increase in Probability of anomaly detection
2	Li, Jiahao, et al.	Novel Methods for Smart Grid Intrusion Detection System Using Feature Selection Based on Improved Gravitational Search Algorithm.	Gravitational Search Algorithm	More accurate Feature selection
3	Ashraf, Javed, et al.	Novel deep learning-enabled LSTM autoencoder architecture for discovering anomalous events from intelligent transportation systems.	LSTM Autoencoder	99% Accuracy on cyberattack detection
4	Javaid, Ahmad, et al.	A deep learning approach for network intrusion	Sparse Autoencoder	79.1%accuracy

		detection system.		
5	Kenyon, Anthony, Lipika Deka, and David Elizondo.	Are public intrusion datasets fit for purpose characterising the state of the art in intrusion event datasets.	-	Little consistent information critically assessing full range of public datasets.
6	Wang, Wei, et al.	HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection.	One Hot encoding	Network data used as input features directly
7	Siddiqi, Murtaza Ahmed, and Wooguil Pak.	Tier-based optimization for synthesized network intrusion detection system.	P Zigzag algorithm	Lower anomaly detection when high dimensional datasets used
8	Pajouh, Hamed Haddad, et al.	A two-layer dimension reduction and two-tier classification model for anomaly-based intrusion detection in IoT backbone networks.	Two Layer Dimension Reduction and Two Tier Classifcation	Accurate detection of anomalies

9	Leevy, Joffrey L., and Taghi M. Khoshgoftaar.	A survey and analysis of intrusion detection models based on cse-cic-ids2018 big data.	CICIDS2018 dataset	Class imbalances was concluded
10	Akbani, Rehan, Stephen Kwek, and Nathalie Japkowicz.	Applying support vector machines to imbalanced datasets.	Support Vector Machines	Classifiers default to majority class
11	Ahmed, Mohiuddin, and Abdun Naser Mahmood.	Novel approach for network traffic pattern analysis using clustering-based collective anomaly detection.	K-Means algorithm variant	Higher accuracy in terms of un labelled data
12	Wisanwanichthan, Treepop, and Mason Thammawichai.	A double-layered hybrid approach for network intrusion detection system using combined naive bayes and SVM.	double-layered hybrid approach	88.97%in accuracy, 90.57%inthe F1score, 88.17%inthe precision, and 93.11%in detection with an 11.82%false alarm ratio
13	Li, Yinhui, et al.	"An efficient intrusion detection system based on support vector machines and gradually feature removal method.	Support Vector Machine	The proportion of the transportation features of target host is high

14	Idhammad, Mohamed, Karim Afdel, and Mustapha Belouch.	Semi - supervised machine learning approach for DDoS detection.	Sliding Window algorithm	Achieved balanced dataset
15	Arisdakessian, Sarhad ,et al.	A survey on IoT intrusion detection: Federated learning, game theory, social psychology, and explainable AI as future directions.	Genetic Algorithm	Higher Detection Rate
16	Sabev, Sabi I.	Integrated Approach to Cyber Defence: Human in the Loop. Technical Evaluation Report.	Four Stage Algorithm	Higher Detection Rate
17	DCunha, S. D.	Is AI Shifting The Human-In-The-Loop Model In Cybersecurity?	Network Traffic Analysis	Efficient Traffic Management
18	Mijalkovic, Jovana, and Angelo Spognardi.	Reducing the false negative rate in deep learning based network intrusion detection systems.	Deep Neural Networks	Differences in accuracy measures between datasets

19	Al-A'araji, Nabeel H., Safaa O. Al-Mamory, and Ali H. Al-Shakarchi.	Classification and clustering based ensemble techniques for intrusion detection systems: A survey.	Bagging	Accurate intrusion detection
20	Aburomman, Abdulla Amin, and Mamun Bin Ibne Reaz.	A survey of intrusion detection systems based on ensemble and hybrid classifiers.	Hybrid classifier	Accurate intrusion detection

Many Intrusion Detection Systems (IDSs) have been created such that they combine features of machine learning hybridism to achieve good performance and high accuracy. As an instance, Mora-Gimeno et al. built a two-phase modular architecture whose main advantage is the improvement of detection using different techniques to build a combined comprehensive view of the application's behavior. But this method faces challenges of computational complexity and scaling.

The Gravitational Search Algorithm (GSA) proposed by Li et al. for feature selection is a mass-based attraction mechanism, but it has a slow convergence rate and is also easily caught in the local optima of a high-dimensional space. Although it is high accurate, its processing speed is slow due to the non-parallel nature of the LSTMs.

The Sparse Autoencoder-based IDS models reported an accuracy of 79.1% on the NSL-KDD dataset. A major problem of the dataset is that, being old, it suffers from class imbalance, both of which greatly hinder real-life applications. While some other methods are either converting network traffic in terms of images for anomalies detection, these techniques hit hitches as regards the detection rates in real-time and reduction of the influence from high dimensional data. Pajouh et al. approached a two-layered dimension reduction and classification model that combined Naive

Bayes and k-NN but only tested it on NSL-KDD, which raises doubts regarding generalization.

Even more challenges are posed by datasets such as CICIDS2018: overfitting, class-imbalance tendencies, and inefficient data cleansing all directly impact IDS performance. Class imbalance also causes biased outputs from classifiers that are of no use in the detection of minority classes. This is Ahmed and Mahmood's semi-supervised strategy which balances both labeled and unlabeled data sets to improve the detection accuracies, addressing the high false alarm rates of unsupervised learning. The double-layer hybrid, the combination of Naive Bayes and SVM, by Wisanwanichthan et al., was able to achieve high accuracy detection but finally had to depend on the quality of training data, making it less adaptable for emerging threats. Idhammad et al. also recommended SMOTE for oversampling the minority classes in order to balance the datasets. However, they warned that too much oversampling would lead to an increase in false alarms.

Most AI-based IDS models, except decision forests, may not use aggregated decomposition techniques in final decision making and suffer from quite high rates of false positives and negatives, making a very strong need for ensemble learning methods to enhance systems performance and also reduce security alerts faced in organizations.

3. PROPOSED SYSTEM

3.1. Methodology

Data Preprocessing:

The method starts from the KDD dataset, which is a reference dataset commonly employed to test IDS. Even if it is rich, the dataset has serious class imbalance in the form of the sparse occurrence of uncommon attacks like Remote-to-Local (R2L) and User-to-Root (U2R) against the common Normal and Denial-of-Service (DoS) entries.

The class imbalance issue is addressed through a GAN model. Generative Adversarial Networks (GANs) are two neural networks — a generator and a discriminator — which play a minimax game wherein the generator produces fake samples and the discriminator estimates their validity. Here, the GAN is trained to generate high-quality synthetic instances of the minority classes (i.e., R2L and U2R). The result of this step is an enriched dataset consisting of both existing and synthetic samples and thereby minimizing the imbalance in class distribution.

After augmentation, the data is assessed for quality. This phase guarantees that synthetic samples are statistically similar to actual data points and have meaningful feature distributions. Metrics like Frechet Inception Distance (FID), t-SNE visualization, and classification confidence thresholds are employed to ensure the realism of synthetic data.

Data Grouping and Encoding:

Once successfully augmented, the data is split into two logical groups for further processing:

Group 1: Comprises all major attack classes — DoS, Probe, R2L, U2R — and Normal traffic.

Group 2: Targets most difficult minority classes — R2L and U2R — and Normal traffic.

This grouping is in service of the layering of architecture. Group 1 goes to the first layer classifier, and Group 2, with the hard-to-detect classes, is treated by the second layer to further improve the classification.

Prior to classification, the two groups both go through normalization and one-hot encoding. Normalization is responsible for ensuring numeric feature values are within a universal scale, facilitating model convergence. One-hot encoding converts categorical attributes into binary vectors, which is required by machine learning algorithms such as Naive Bayes and XGBoost that work with numerical input.

Balancing and Downsampling

Although the GAN has treated imbalance in the entire dataset, downsampling is selectively performed on Group 2 to balance class distributions further. This is a necessary step in avoiding XGBoost from becoming biased towards the majority class, particularly when encountering residual imbalance on re-encounter after augmentation. Downsampling entails reducing the number of instances of the majority class (e.g., Normal) to be equal to that of minority instances (R2L, U2R).

Classifier Construction

Layer 1: Naive Bayes Classifier

Group 1 is utilized to train an Naive Bayes (NB) classifier, a probabilistic machine learning algorithm owing to Bayes' Theorem which has high (naive) feature independence assumptions. NB is fast and efficient, making it perfect for early classification. It captures well dominant attack types such as DoS and Probe and can effectively send unclear or misclassified samples to the next layer.

Layer 2: XGBoost Classifier

Group 2 is passed to a Gradient Boosting classifier in XGBoost, one of the best-performing ensemble methods for tabular data. XGBoost follows additive training using gradient descent over loss functions in an effort to optimize performance via methods such as column subsampling, regularization, and tree pruning. This layer

aims at separation between Normal, R2L, and U2R categories that often lag behind with worse detection rates. By separating this task, Layer 2 refines the classification of past uncertain samples more precisely.

Integrated Evaluation and Feedback

After both layers are trained and tested, their outputs are merged and measured using performance metrics such as accuracy, precision, recall, F1-score, and ROC-AUC. Particular attention is given to F1-score for minority classes due to the original imbalance of the dataset. In addition, quality feedback loops measure the effect of GAN-generated data by comparing model performance with and without augmentation, thereby measuring the real benefit of generative improvement.

3.1.1. Architecture Diagram

The system architecture of the proposed system is a complex double-layered hybrid system that has been designed specifically to overcome the limitations related to traditional Intrusion Detection Systems (IDS), particularly in terms of unbalanced datasets as well as low detection rates against rare attack types. It relies on an improved Double-Layered Hybrid Approach (DLHA) which replaces Support Vector Machine (SVM) in the second layer with eXtreme Gradient Boosting (XGBoost), and employs Generative Adversarial Networks (GANs) in order to execute smart data augmentation. The following subsections present a detailed step-by-step description of the architecture as depicted in Figure 3.1.

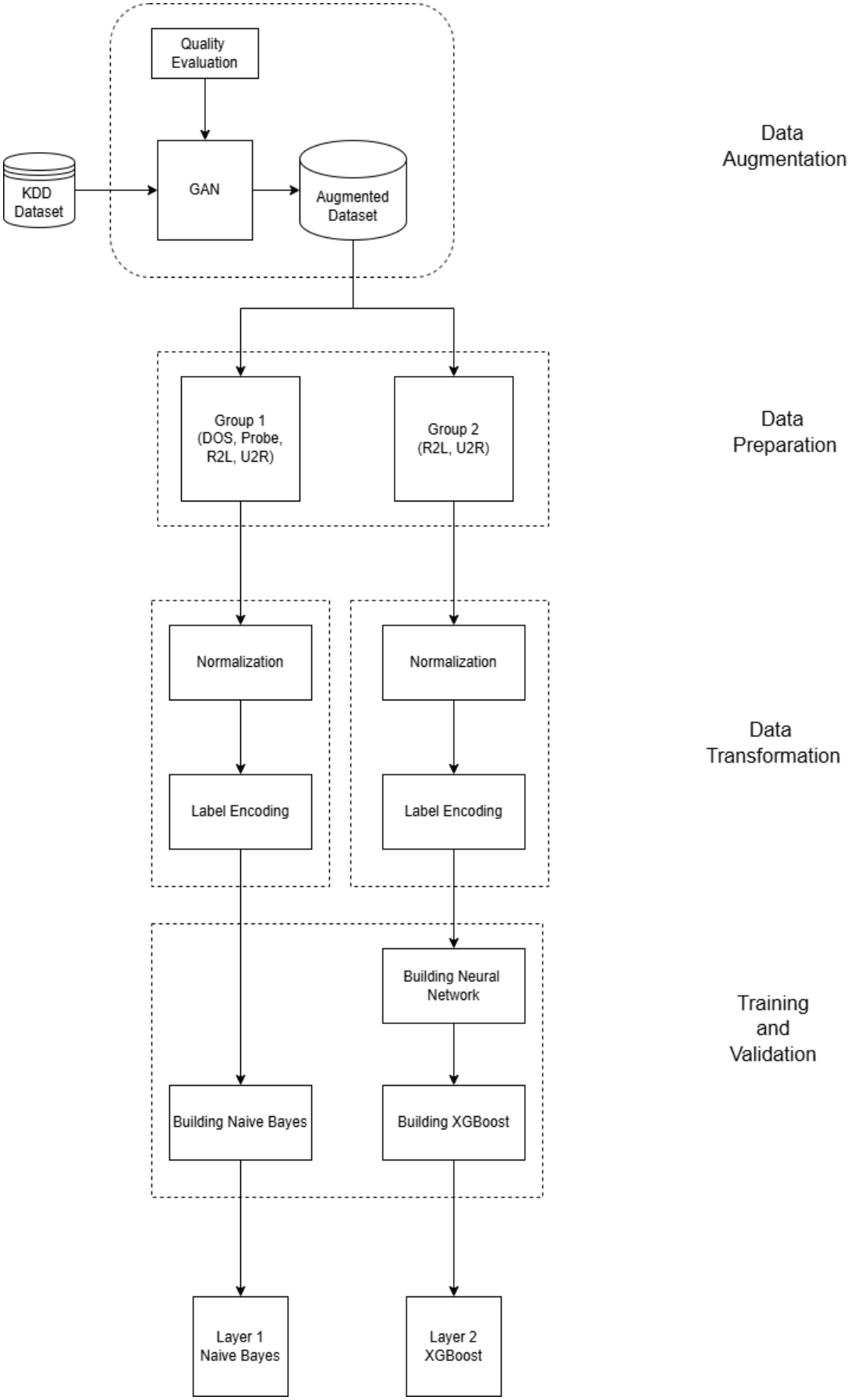

Figure 3.1. Architecture Diagram

3.1.2. Functional Modules

Data Augumentation

The process started with a basic Generative Adversarial Network (GAN) to address the class imbalance in the KDD dataset concerning rare attack types such as Remote-to-Local (R2L) and User-to-Root (U2R). The vanilla GAN was able to generate synthetic samples with minor improvements but faced severe mode collapse and unstable training that limited all diversity and quality of the generated samples for the minority class. Addressing this with the usage of Conditional Tabular GAN (CT-GAN) reflected much improvement over the structured dataset, but it did not solve its bottleneck of misrepresentation of the minority distributions.

This was followed by employing a Wasserstein GAN (W-GAN) for enhancing training stability and limiting mode collapse by adopting the Wasserstein loss function. Compared to the previous architecture, W-GAN had improved gradient behavior and convergence; however, it still majorly lacked stringent control of sample quality. The final improvement came with W-GAN, which included the gradient penalty, which even further enforced Lipschitz continuity and stabilized the training process manyfold. This variant produced high-quality, diverse synthetic samples of the minority classes, which effectively enlarged the dataset and minimized imbalance.

To validate the quality of generated samples, there are Frechet Inception Distances (FID) for the distribution similarity assessment, t-SNE visualizations for feature overlaps, and classification confidence thresholds for realism test. This battery of tests guarantees that the synthetic data appreciably add value to the aggregate performance of a detection system.

Data Preparation

Upon completion of augmentation, the dataset is logically partitioned into two groups. Group 1 contains all the primary classes — DoS, Probe, R2L, U2R, and Normal traffic — used to train the first-layer classifier. Group 2 separates the harder minority classes of R2L and U2R along with Normal traffic, which is made available to the second-layer classifier for training. This grouping facilitates a

layered approach to classification where more difficult distinctions are assigned to a particular model.

Before the model is trained, preprocessing is accomplished to ensure that both groups are consistent numerically. All numerical features are normalized to a common scale to support efficient model training and convergence. Label encoding is applied to categorical features by transforming each unique category into an associated value. This encoding ensures compatibility with machine learning algorithms such as Naive Bayes and XGBoost that typically require numerical input.

Data Transformation

Once the categorization is accomplished each category passes through these activities: encoding, normalization and integrity of information. Protocol_type, service and flag columns to be transformed into numbers and later trained with these numbers. Encoding is done both on the train data and test data to make sure that the result is across all datasets. Thereafter numeric data are standardized and scaled using StandardScaler classifier. The resulting standard numeric data will have mean value equal to zero and standard deviation of one. This scaling process is necessary because it ensures that large feature values do not overpower the model. Input data for machine learning model includes numerical values.

The original purpose of GAN was to have this narrow approach for any further level of imbalance in Group 2. Selective downsampling across the majority class in Group 2, mainly reducing the number of Normal samples, has been employed to overcome this. Therefore, this step would prevent models such as XGBoost from leaning too much towards the dominant patterns but treat the R2L and U2R samples fairly.

Training and Validation

The architecture contains a two-layer classification. In the first layer, the Naive Bayes classifier is trained using Group 1. This model is an efficient large-scale classification method based on Bayes' Theorem and works primarily on the

independence-of-features assumption. It casts the classification task fairly well on the popular classes such as DoS and Probe while marking ambiguous cases that may require further attention.

Group 2 flows into the second layer, which uses XGBoost as the classifier. XGBoost, a powerful implementation of gradient boosting, improves classification performance on the more challenging classes, R2L and U2R. To achieve better performance, it employs regularization, tree pruning, and column subsampling. This layer is especially designed to fix the problems triggered by the first layer so that it can ensure better detection of the rare attacks.

The outputs of both classifiers are merged and evaluated using standard metrics such as accuracy, precision, recall, F1-score, and ROC-AUC. A major emphasis is placed on the minority classes' F1-score for tracking the adaptation of the model concerning the original imbalance. A feedback system is implemented to compare performance with and without GAN augmentation, thus helping to quantify the actual effect of synthetic data in detection accuracy.

4. EXPERIMENTAL SETUP & RESULTS

4.1. System Requirements

Development Platform

Kaggle Notebooks serve as the main development and execution platform for this project. It is usually cloud-based Jupyter, which comes loaded with most of the essential libraries and hardware accelerators; hence, there is no need for heavy local installation and first-time access available on both GPUs and TPUs free, making it the most efficient platform for deep learning models such as GANs.

Local System Compatibility

Even though Kaggle does most heavy-lifting computing operations, a local system should have the minimum required compatibility to run basic machine learning workflows. This allows small-scale experimentation, data preprocessing, or debugging to be done offline if necessary and prevents the need for constant connection to the internet or access to cloud resources.

Internet Connectivity

Since all code development, dataset upload/download, model training, and notebook execution are done on cloud, online connectivity is very important. Also, that training with GANs and training data can take up a lot of memory, hence the higher speeds, preferably 50 Mbps or more, really cut down on the wait time when transferring data and uploading model checkpoints.

Browser Support

A modern web browser is needed to access the Kaggle interface. Recommended options include the latest version of Google Chrome, Mozilla Firefox, or Microsoft Edge. These browsers support the advanced JavaScript and interactive rendering

required by Jupyter Notebooks so that the user can work without compatibility issues.

Storage Considerations

While Kaggle does store online data sets and notebook outputs, backup copies of any vital code, models, or outputs should be on a local system or cloud drive. For this, keeping locally at least 20-50 GB minimum free space is handy, especially when working with the intermediate model checkpoints or versions.

4.1.1. Software Requirements

A current, supported operating system-Windows 10/11, Ubuntu 20.04 LTS, or macOS Monterey (12) and above should be installed on the local environment for compatibility with current development tools and GPU drivers. Technically, Kaggle notebooks can run on Windows 8 through the browser, but its obsolete security patches and driver support make it illogical for local deep-learning execution.

Python 3.7, 3.8, or 3.9 is recommended as a runtime, with Python 3.8 as the default on Kaggle. In-between, these versions allow access to new language features-like better typing, async improvements, and performance optimizations-while being compatible with larger library stacks. A virtual environment (venv or conda) helps keep dependencies isolated and prevents version conflicts for different projects.

Among all libraries, these are essential for data science computation: Numerical Python (NumPy) is best for array operations and linear algebra routines; Pandas is excellent for flexible data manipulation and I/O; scikit-learn contains a well-designed set of classical machine learning algorithms and preprocessing tools; imbalanced-learn implements sampling strategies to address class imbalance; and

XGBoost is a high-performance gradient boosting library, particularly for heavy lifting on tabular data.

For GAN training and deep learning, you want a very competent framework. TensorFlow ≥ 2.5 is recommended for TPU usage since it has excellent integration with Google accelerator hardware and high-level APIs such as Keras. In contrast, PyTorch ≥ 1.9 is seen as the best choice for traditional GPU-based implementation. The features that have led to its popularity are dynamic computation graphs, a rich model zoo, and a vibrant community.

Exploratory data analysis and monitoring long-running tasks for model development and tinkering are done in a notebook setting, such as Kaggle's integrated JupyterLab or locally installed JupyterLab. These scientific computing environments allow interactive code execution and inline visualization, and enhance collaborative work through shareable notebooks. Optional utilities like Matplotlib and Seaborn for visualization, and Tqdm for progress reporting, facilitate exploratory analysis and monitoring of long-running tasks.

4.1.2. Hardware Requirements

The Kaggle Notebook Environment allocates 2 virtual CPUs and 16 GB of RAM in a shared configuration for use in the cloud kernel; 20 GB of persistent storage is allocated to datasets, while outputs and cache get only 5 GB of storage. The notebook allows a choice between hardware accelerators in form of either an NVIDIA Tesla P100 GPU with 16 GB of VRAM or Google TPU v3. All drivers, CUDA/cuDNN versions, and TPU runtimes are preconfigured, leaving the user time to devote entirely to the development of the model.

For cases where experiments or data preprocessing are done locally, a machine with at least 16 GB RAM (preferably 32 GB) and an eight-core CPU such as Intel i7-

9700 or AMD Ryzen 7 3700X is recommended. A fast NVMe SSD of 512 GB or more will be great for quick data loading and model checkpointing. If training GANs locally, an NVIDIA GPU such as GTX 1660 Ti (6 GB VRAM) or RTX 2060 (8 GB VRAM) with up-to-date CUDA 11.x and cuDNN 8.x drivers will significantly reduce the time for trainings; RTX 3060 or RTX 3070 (12 GB+) is an ideal option.

4.2. Description

NSL-KDD dataset, which is considered as one of the most popular standards for intrusion detection system testing, has three sets: KDDTrain+, KDDTrain+_20Percent, and KDDTest+. The set KDDTrain+_20Percent represents 20% of the samples in KDDTrain+ for the same class distribution ratio. Training process occurs on both the size KDDTrain+ full dataset and the smaller KDDTrain+_20Percent data. In this way, performance differences due to the change in training data size are evaluation; how the algorithm performs and is robust for low data is helpful.

The NSL-KDD dataset contains 3 subsets- KDDTrain+, KDDTrain+_20Percent, and KDDTest+, which have been generally used as benchmarks for intrusion detection systems. The KDDTrain+_20Percent subset contains 20% of the instances from the KDDTrain+, maintaining the same class distribution ratio. The full-sized KDDTrain+ and smaller KDDTrain+_20Percent datasets undergo training process. That way, performance differences can be viewed when the training data size changes, indicating how well the algorithm performs and is robust for small amounts of data.

The KDDTrain+ set contains KDDTrain+, KDDTrain+_20Percent, and KDDTest+, the three subsets comprising the NSL-KDD dataset, which is up to now among the most popular standards that have been used to evaluate intrusion detection systems. The set KDDTrain+_20Percent has 20% of instances in the KDDTrain+ dataset, preserving the same ratio class distribution. The full-sized KDDTrain+ dataset and the smaller KDDTrain+_20Percent data have training process running on them. This allows evaluation of performance differences in the

variation of size of training data on how the algorithm performs well and its robustness for low data. The columns in the KDDTrain+ are stated in the Figure 4.1 below.

Group	No.	Feature Name	Description	Type
Intrinsic Features	1	Duration	Length of time of the connection	Continuous
	2	Protocol Type	Type of protocol used in the connection	Categorical
	3	Service	Destination network service	Categorical
	4	Flag	Status of the connection	Categorical
	5	Src Bytes	Number of data bytes transferred from source to destination	Continuous
	6	Dst Bytes	Number of data bytes transferred from destination to source	Continuous
	7	Land	1 If source and destination IP addresses are equal	Discrete
	8	Wrong Fragment	Total number of wrong fragments in this connection	Discrete
	9	Urgent	Number of urgent packets	Discrete
Content Features	10	Hot	Number of hot indicators in the content	Continuous
	11	Num Failed Logins	Count of failed login attempts	Continuous
	12	Logged In	Login Status, 1 if successful	Discrete
	13	Num Compromised	Number of "compromised" conditions	Continuous
	14	Root Shell	1 if root shell is obtained; 0 otherwise	Discrete
	15	Su Attempted	1 if "su root" command attempted or used	Discrete
	16	Num Root	Number of "root" accesses	Continuous
	17	Num File Creations	Number of file creation operations in the connection	Continuous
	18	Num Shells	Number of shell prompts	Continuous
	19	Num Access Files	Number of operations on access control files	Continuous
	20	Num Outbound Cmds	Number of outbound commands in an ftp session	Continuous
	21	Is Hot Logins	1 if the login belongs to the "hot" list	Discrete
	22	Is Guest Login	1 if the login is a "guest" login	Discrete
Time-based Features	23	Count	Number of connections to the same dst host in the past two seconds	Continuous
	24	Srv Count	Number of connections to the same service in the past two seconds	Continuous
	25	Serror Rate	% connections have activated the flag (4) s0, s1, s2 or s3 from (23)	Continuous
	26	Srv Serror Rate	% connections have activated the flag (4) s0, s1, s2 or s3 from (24)	Continuous
	27	Rerror Rate	% connections have activated the flag (4) REJ from (23)	Continuous
	28	Srv Rerror Rate	% connections have activated the flag (4) REJ from (24)	Continuous
	29	Same Srv Rate	% connections to the same service from (23)	Continuous
	30	Diff Srv Rate	% connections to different services from (23)	Continuous
	31	Srv Diff Host Rate	% connections to different dst machines from (24)	Continuous
Host-based Features	32	Dst Host Count	Count for destination host	Continuous
	33	Dst Host Srv Count	Srv-count for destination host	Continuous
	34	Dst Host Same Srv Rate	Same-srv-rate for destination host	Continuous
	35	Dst Host Diff Srv Rate	Diff-srv-rate for destination host	Continuous
	36	Dst Host Same Src Port Rate	Same-src-port-rate for destination host	Continuous
	37	Dst Host Srv Diff Host Rate	Diff-host-rate for destination host	Continuous
	38	Dst Host Serror Rate	Serror-rate for destination host	Continuous
	39	Dst Host Srv Serror Rate	Srv-serror-rate for destination host	Continuous
	40	Dst Host Rerror Rate	Rerror-rate for destination host	Continuous
	41	Dst Host Srv Rerror Rate	Srv-rerror-rate for destination host	Continuous

Figure 4.1. Description of column in KDDTrain+

The dataset includes five primary classes of network activity:

1. **Denial of Service (DoS)**
2. **Probe**
3. **Remote-to-Local (R2L)**
4. **User-to-Root (U2R)**
5. **Normal**

Denial of Service (DoS)

A Denial of Service (DoS) attack is a tactic that is used to disrupt the availability of services, systems, or networks by bombarding them with waves of unnecessary requests or great amounts of malicious traffic. The principle of the attack falls on the depletion or exhaustion of system resources, such as CPU, memory, and bandwidth, to make it unresponsive to legitimate users. DoS attacks are possible because of vulnerabilities within the system, weaknesses in implemented protocols, or misconfigurations with little or no access rights. Those attacks may happen, for example, against web servers, application servers, routers, or the entire network. Deny normally theft of data, but due to its capability of bringing down an essential service, it is highly disruptive, particularly on online platforms, which survive on very high uptime and responsiveness. They could also be resorted to diversions to mask other kinds of malicious activity by an attacker.

Example Scenario:

An attacker sends many TCP SYN packets to a web server without completing a handshake. The server acknowledges waits for acknowledgment and, thus, allocates resources, eventually exhausting this capacity and denying access to real users.

GitHub Attack (2018)

In February 2018, GitHub was hit by one of the largest DoS attacks ever recorded at the time, peaking at 1.35 terabits per second. The attackers exploited memcached servers to amplify the volume of traffic. Although GitHub managed to recover quickly thanks to automated defenses, the attack highlighted the growing severity

of DoS attacks and how easily misconfigured servers can be abused for large-scale disruption.

Probe

Probe attacks are gathering information about the structure, services, and vulnerabilities in the network or system through scanning. By identification of probes, attackers can open certain ports, find out which services are running, determine software versions, or discover misconfigurations. Although probe attacks are not destructively immediate, they are at times precursors to more serious intrusions.

Some of the common probe techniques include port scanning, ping sweeps, and vulnerability scanning. The resultant intelligence now considerably prepares the stage for the attacker to launch more pointed attacks such as buffer overflows, password guessing, or privilege escalation. They can probe targets with automated tools which scan very large address spaces quickly and are typically not detected unless network monitoring is in place. Organizations then consider probing a major threat because it is a show of an opponent trying to pry into and breach their systems.

Example Scenario:

An attacker executes an Nmap scan against a corporate network to verify which machines have SSH or FTP ports open that would represent potential vulnerabilities for them when they plan to intrude into the network in the future.

Stuxnet Reconnaissance Phase (2009–2010)

Before the destructive payload of the Stuxnet worm was deployed to sabotage Iran's nuclear centrifuges, it first engaged in extensive probing. The malware stealthily gathered detailed information about Siemens PLCs (Programmable Logic Controllers) in the Natanz facility. This probing allowed attackers to design precise and targeted commands that caused physical damage to the centrifuges. Public sources confirm the scope of reconnaissance but do not specify the exact volume of

probes, underscoring the stealth and sophistication of Stuxnet's intelligence gathering.

Remote-to-Local (R2L)

Remote-to-local (R2L) attacks refer to attempts by an attacker to gain unauthorized access to a system via a network without any local access. This access usually involves obtaining local user privileges by manipulating weak authentication mechanisms, poorly configured services, or exploiting unpatched vulnerabilities. R2L attacks are especially significant because they present an opportunity for adversaries to attack the perimeter or network from outside.

Such attacks may involve password guessing, exploitation of vulnerabilities remotely in applications such as ftp or email servers, or phishing involving tricking the user to give his credentials. Unlike DoS or Probes, R2L remains subtle and purposed at the establishment of long-term unauthorized access, which becomes foundational for acquiring higher-level breaches eventually.

Example Scenario:

An attacker creates a crafted email message and sends it to an employee, and when clicked on the link of this email, it exploits the flaw in the mail web service of the company and gives a local shell on the machine of the victim to the attacker.

SolarWinds Hack (2020)

The SolarWinds cyberattack, attributed to a Russian APT group, involved Remote-to-Local techniques. Attackers inserted malicious code into SolarWinds' Orion software update, which was then installed by over 18,000 organizations, including U.S. government agencies. Once inside, the attackers moved laterally within networks, gaining unauthorized access to sensitive internal systems using valid credentials—a hallmark of a sophisticated R2L attack.

User-to-Root (U2R)

User-to-Root (U2R) attacks allow the perpetrator with minimal user access privileges, typically limited to a standard user, to elevate themself or themselves to superuser or root access. U2R attacks exploit local vulnerabilities such as buffer overflows, misconfigured permissions, or kernel flaws. U2R attacks are more difficult to implement than R2L attacks because a user must have access to a local account to perform such attacks; however, once successful, they provide unrestricted access to the entire system. Such unrestricted access by the attacker may include installing backdoor access, cleaning up logs, changing configurations, or introducing malicious programs. U2R attacks are quite dangerous as they can compromise an entire system and can be easily used in an advanced persistent threat (APT) where stealth and control are critical.

Example Scenario:

A non-specific intrusive user runs exploit on a Linux system that makes use of a buffer overflow in the sudo command to execute commands stealing root access and bypassing the system restrictions.

Linux Sudo Vulnerability (Baron Samedit, 2021)

In 2021, a critical U2R vulnerability in the sudo utility (CVE-2021-3156), dubbed "Baron Samedit," was discovered. This flaw allowed any local user on a Unix-like system to escalate privileges to root by exploiting a heap overflow. The vulnerability went undetected for nearly a decade and affects virtually all default sudo installations worldwide, underlining the danger of U2R exploits in widely used software.

Normal

The normal traffic is legitimate expected network behavior within computer networks. This includes normal browsing, sending messages, transferring files, logging into a system, or accessing a database. These actions exhibit normal patterns of usage, conform to security policies, and do not attempt to breach weaknesses in the system.

Normal data will help establish the basic profile for Intrusion Detection Systems (IDS), which then can recognize anomalies, indicating possible malicious activity. It is often quite difficult to distinguish intelligent attacks from normal behavior, especially if the attacker has learned to imitate legitimate actions. It is also important to precisely model normal traffic to limit false positives and improve efficiency of IDS.

Example Scenario:

An employee has logged into his work account using SSH, checked emails, and uploaded a project document to the secure file server of the company-these activities are normal activity within the network.

Edward Snowden NSA Breach (2013)

Edward Snowden, a systems administrator at the NSA, used normal system access privileges to download and exfiltrate over 1.7 million classified files without raising immediate suspicion. His activities appeared as legitimate admin tasks, which is why they evaded detection for a considerable period. This case shows how normal behavior can be exploited to mask severe internal breaches.

The percentage distribution of these classes within the dataset is as follows:

Table 4.1. Distribution of Classes in Original KDDTrain+

Category	Class	Samples	Percentage
Frequent	DoS	45927	37.34
Frequent	Probe	11656	9.47
Rare	R2L	995	0.8
Rare	U2R	52	0.00042
Normal	Normal	67343	54.76

Table 4.2. Distribution of Classes in Enchaned KDDTrain+

Category	Class	Samples	Percentage
Frequent	DoS	65930	24.77
Frequent	Probe	11656	4.40
Rare	R2L	81003	30.45
Rare	U2R	40053	15.06
Normal	Normal	67343	25.32

This very much creates quite a pool and classes that imbalance a lot especially with R2L and U2R attacks while being ineffective when it comes to the issues of training and detection. In lieu of that, we have already used Generative Adversarial Networks (GAN) in augmenting data. The synthesis of further entries for scarce attack types provides a means of reducing the imbalance of classes as seen in Table 4.1 and Table 4.2 where the imbalance in Rare category attacks was reduced, enabling the model to concoct better learning using the features of these underrepresented classes.

Unseen Attack Classes

KDDTest+ contains 17 minor classes of new attacks such that apache2, httptunnel, mailbomb, mscan, named, processtable, ps, saint, sendmail, snmpgetattack, snmpguess, sqlattack, udpstorm, worm, xlock, xsnoop, xterm, and so on are nonexistent in their corresponding training datasets. Thus, the model becomes important to generalization in attacks by evaluating it under conditions of previously unseen instances.

Moreover, it is important to note that two more minor attack classes, namely spy and warezclient, have been included. They are present in the training datasets but not available in any of the testing data. Therefore, this scenario can be addressed in terms of robustness and adaptability of the model towards upcoming changes in the attack types. Stronger detection with such infrequent and unseen attacks would make the entire intrusion detection system to be effective and scale in nature through the addition to the datasets up by GANs.

4.3. Results and Test Analysis

Table 4.3. Overall Model Performance on KDDTest+

Model	Accuracy	F1 Score	Precision	Recall	FAR
Original DLHA (NB + SVM)	85.09%	79.98%	76.08%	85.09%	1.33%
Decision Tree	85.66%	81.64%	87.45%	85.66%	1.07%
Random Forest	86.71%	81.61%	82.52%	86.71%	1.18%
Single-layer XGBoost	86.08%	80.92%	76.70%	86.08%	1.29%
Proposed DLHA (ours)	85.38%	80.20%	76.29%	85.38%	1.32%

According to Table 4.3, the proposed GAN-augmented DLHA strategy obtains a good balance among all metrics, recording a competitive accuracy of 85.38% and F1-score of 80.20%. Though slightly lower than Random Forest and Decision Trees in terms of accuracy, it ranks very well across all evaluation metrics. With improved detection of minority classes, its performance reiterates its superiority over the other models.

Class-wise Detection Rate

Table 4.4. Detection Rate per Attack Type

Class	Original DLHA	Proposed DLHA
DoS	92.4%	92.60%
Probe	90.8%	97.96%
R2L	74.67%	89.16%
U2R	55.0%	60.42%
Normal	85.3%	92.0%

The improvements in detection rates of the proposed DLHA model concerning all attack types are highlighted in Table 4.4. For instance, minority class detection gains are impressive for R2L (89.16%) and U2R (60.42%) as compared to original DLHA. Improvements achieved across all the categories certainly witness the superiority of our model over the original across all parameters stated above. The improvement is reflected in the graph (Figure 4.2.) shown below the table as well.

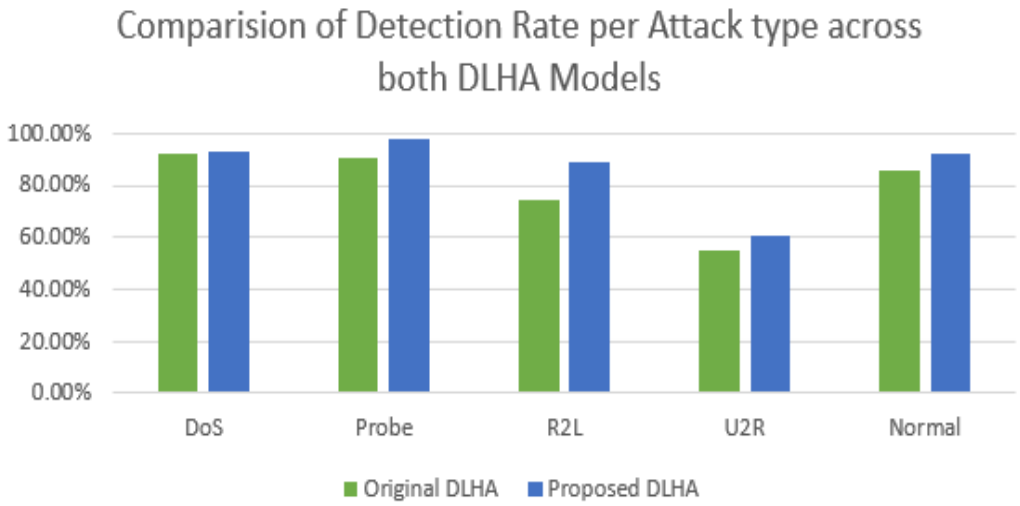

Figure 4.2. Graph comparing Performance Metrics between two DLHA models

Evaluation Metrics

Accuracy.

Accuracy shows how often the model predicts correctly out of all it's predictions. It's a measure of the model's overall performance in classification tasks. The accuracy of a model (Acc) is calculated using the following formula:

Acc = (All predictions made/Total number of Correct Predictions) × 100%

Where:

- Correct predictions are the correctly classified instances by the model
- Total predictions are the total classified instances by the model

Recall.

Recall (true positive rate) or sensitivity, measures the model's ability to find all the positive cases in the dataset. It's the proportion of correctly identified positives out of all the actual positive cases.

Recall = Correct Positives / (Correct Positives + Wrong Negatives)

Precision.

It is the ratio of correctly predicted positive cases to all cases predicted as positive. It shows how accurate the model is when it predicts something as positive.

$$\text{Precision} = \text{Correct Positives} / (\text{Correct Positives} + \text{Wrong Positives})$$

F1 Score.

The F1 score is like a balanced average of how accurate and thorough a model is in finding the right answers, especially when some answers are rare. It combines precision and recall into one number.

$$\text{F1 Score} = 2 \times \text{Precision} \times \text{Recall} / (\text{Precision} + \text{Recall})$$

False Alarm Rate.

False Alarm Rate (FAR) is the proportion of normal instances that are incorrectly classified as attacks. It is calculated as:

$$\text{FAR} = \text{Wrong Positives} / (\text{Wrong Positives} + \text{Correct Negatives})$$

5. CONCLUSION AND FUTURE SCOPE

Employing XGBoost, dimensionality reduction, and an additional dataset, this project effectively presents the creation of a network intrusion detection system that is refined and capable of providing enhanced and advanced detection of cyberattacks. Most of the traditional systems face the challenges of having imbalanced datasets, which makes the detection of the rare types of attacks less efficient.

Combining modern machine learning approaches with various specific and customized data preparation and augmentation methods, this project seeks to enhance both intrusions detection accuracy and its operational capability in practice. Network traffic data is processed in the system by feature extraction followed by feature normalization.

In order to overcome the problem of underrepresentation, class imbalance, data augmentation strategies such as the synthetic minority oversampling technique are employed on the training data. Augmentation ends up balancing the dataset as it brings to the fore, the representations of such infrequent types of attacks as remote-to-local and user-to-root, which are seen to be little in the raw data.

More complex and more frequent cyber-crimes demand better and adaptive Intrusion Detection Systems (IDS). Today's systems like the Double-Layered Hybrid Approach (DLHA), outperforming detection rates for high-frequency attacking types like Denial of Service (DoS) and Probe, are still strong for even rare ones like R2L and U2R. However, since the attacks change rapidly, static detection is no longer a viable option.

Further, IDS designs for the future will need to include self-adaptive capabilities that would allow them to simultaneously evolve with the threats. Such reinvention mechanisms would make use of learning reinforcement or online learning mechanisms whereby real-time detection would continually modify the model based on newly discovered attack patterns.

It will always be the right clicks, as the goal for an IDS would continue to be optimized for accuracy. Although DLHA has an impressive accuracy rate, the high False Alarm Rate (FAR) indicates that there is still room for improvement. Further,

deep neural networks or transformer architectures can improve the ability of the system to detect hardly perceptible attacks without increasing the FAR.

In addition, Generative Adversarial Networks can be applied for augmenting the dataset, which can solve the class imbalance issue-cause by the rare, synthesize some samples for training the model. The other critical area needing improvement is the use of dynamic techniques for feature selection. Presently practiced techniques like the Intersectional-Correlated-Feature Selection (ICFS) have proven to be useful; however, the performance is highly dependent on the training data. Systems in the future could use adaptive feature selection that would identify and give priority to the most focused relevant features on-the-fly, helping improve detection accuracy and cut down the computational load.

The third area is concerned with making future IDS balance between performance efficiency and accuracy, as this would guarantee scalability in very high-traffic networks. With the help of distributed computing and parallel processing frameworks, such as Apache Spark or TensorFlow, future IDS can easily be deployed in the real world, where they must deal with issues of extremely low latency and extremely high throughput. Hence, addressing the above research issues will lead to the making of future IDS frameworks that will provide strong, scalable, and adaptive solutions to the impacts of evolving cyber threats.

REFERENCES

[1] Mora-Gimeno, Francisco José, et al. "Intrusion detection system based on integrated system calls graph and neural networks." IEEE Access 9 (2021): 9822-9833.

[2] Li, Jiahao, et al. "Novel Methods for Smart Grid Intrusion Detection System Using Feature Selection Based on Improved Gravitational Search Algorithm." 2024 9th International Conference on Automation, Control and Robotics Engineering (CACRE). IEEE, 2024.

[3] Ashraf, Javed, et al. "Novel deep learning-enabled LSTM autoencoder architecture for discovering anomalous events from intelligent transportation systems." IEEE Transactions on Intelligent Transportation Systems 22.7 (2020): 4507-4518.

[4] Javaid, Ahmad, et al. "A deep learning approach for network intrusion detection system." Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS). 2016.

[5] Kenyon, Anthony, Lipika Deka, and David Elizondo. "Are public intrusion datasets fit for purpose characterising the state of the art in intrusion event datasets." Computers & Security 99 (2020): 102022.

[6] Wang, Wei, et al. "HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection." IEEE access 6 (2017): 1792-1806.

[7] Siddiqi, Murtaza Ahmed, and Wooguil Pak. "Tier-based optimization for synthesized network intrusion detection system." IEEE Access 10 (2022): 108530-108544.

[8] Pajouh, Hamed Haddad, et al. "A two-layer dimension reduction and two-tier classification model for anomaly-based intrusion detection in IoT backbone networks." IEEE Transactions on Emerging Topics in Computing 7.2 (2016): 314-323.

[9] Leevy, Joffrey L., and Taghi M. Khoshgoftaar. "A survey and analysis of intrusion detection models based on cse-cic-ids2018 big data." Journal of Big Data 7 (2020): 1-19.

[10] Akbani, Rehan, Stephen Kwek, and Nathalie Japkowicz. "Applying support vector machines to imbalanced datasets." Machine Learning: ECML 2004: 15th European Conference on Machine Learning, Pisa, Italy, September 20-24, 2004. Proceedings 15. Springer Berlin Heidelberg, 2004.

[11] Ahmed, Mohiuddin, and Abdun Naser Mahmood. "Novel approach for network traffic pattern analysis using clustering-based collective anomaly detection." Annals of Data Science 2.1 (2015): 111-130.

[12] Wisanwanichthan, Treepop, and Mason Thammawichai. "A double-layered hybrid approach for network intrusion detection system using combined naive bayes and SVM." Ieee Access 9 (2021): 138432-138450.

[13] Li, Yinhui, et al. "An efficient intrusion detection system based on support vector machines and gradually feature removal method." Expert systems with applications 39.1 (2012): 424-430.

[14] Idhammad, Mohamed, Karim Afdel, and Mustapha Belouch. "Semi-supervised machine learning approach for DDoS detection." Applied Intelligence 48 (2018): 3193-3208.

[15] Arisdakessian, Sarhad, et al. "A survey on IoT intrusion detection: Federated learning, game theory, social psychology, and explainable AI as future directions." IEEE Internet of Things Journal 10.5 (2022): 4059-4092.

[16] Sabev, Sabi I. "Integrated Approach to Cyber Defence: Human in the Loop. Technical Evaluation Report." Information & Security: An International Journal 44 (2020): 76-92.

[17] DCunha, S. D. "'Is AI Shifting The Human-In-The-Loop Model In Cybersecurity?." 2017,

[18] Mijalkovic, Jovana, and Angelo Spognardi. "Reducing the false negative rate in deep learning based network intrusion detection systems." Algorithms 15.8 (2022): 258.

[19] Al-A'araji, Nabeel H., Safaa O. Al-Mamory, and Ali H. Al-Shakarchi. "Classification and clustering based ensemble techniques for intrusion detection systems: A survey." Journal of Physics: Conference Series. Vol. 1818. No. 1. IOP Publishing, 2021.

[20] Aburomman, Abdulla Amin, and Mamun Bin Ibne Reaz. "A survey of intrusion detection systems based on ensemble and hybrid classifiers." Computers & security 65 (2017): 135-152.

APPENDIX

https://github.com/vkssrinivas/Intrusion-Detection-using-GAN

Main Code

 # Step 1: Load & Preprocess Dataset

```
import pandas as pd

from sklearn.preprocessing import LabelEncoder



# Paths

train_dataset_path = "/kaggle/input/merged-dataset-1/MergedDataset.csv"

test_dataset_path  = "/kaggle/input/kddtest/KDDTest.txt"

field_names_path   = "/kaggle/input/fieldsnames/Field Names.csv"



# 1. Load the 41 feature names, then define the train vs test column lists

features = pd.read_csv(field_names_path, header=None).iloc[:,0].tolist()  # length 41

train_cols = features + ["label"]

test_cols  = features + ["label", "attack_id"]  # KDDTest.txt has an extra numeric attack_id column



# 2. Read train (42 cols) and test (43 cols), then drop the extra attack_id

df_train = pd.read_csv(train_dataset_path, header=None, names=train_cols, skiprows=1, low_memory=False)

df_test  = pd.read_csv(test_dataset_path,  header=None, names=test_cols, skiprows=1, low_memory=False)

df_test.drop(columns="attack_id", inplace=True)
```

```
# 3. Label-encode the three categorical features on the union of train+test
categorical_cols = ["protocol_type", "service", "flag"]
encoders = {}
for col in categorical_cols:
    # normalize strings
    df_train[col] = df_train[col].astype(str).str.lower().str.strip()
    df_test[col]  = df_test[col].astype(str).str.lower().str.strip()
    # fit on combined and transform both
    le = LabelEncoder().fit(pd.concat([df_train[col], df_test[col]]))
    df_train[col] = le.transform(df_train[col])
    df_test[col]  = le.transform(df_test[col])
    encoders[col] = le

# 4. Map the string attack-names → integers 1…23 (drop any others)
mapping = {
    'back':1,'buffer_overflow':2,'ftp_write':3,'guess_passwd':4,'imap':5,
    'ipsweep':6,'land':7,'loadmodule':8,'multihop':9,'neptune':10,
    'nmap':11,'perl':12,'phf':13,'pod':14,'portsweep':15,
    'rootkit':16,'satan':17,'smurf':18,'spy':19,'teardrop':20,
    'warezclient':21,'warezmaster':22,'normal':23
}
for df in (df_train, df_test):
```

```
    df["label"] = (
        df["label"]
          .astype(str)
          .str.rstrip('.')    # remove trailing dot in test
          .str.lower()
          .str.strip()
          .map(mapping)
    )
# drop any rows not in 1…23
df_train = df_train[df_train.label.notna()].reset_index(drop=True)
df_test  = df_test [df_test. label.notna()].reset_index(drop=True)
df_train["label"] = df_train["label"].astype(int)
df_test [ "label"] = df_test [ "label"].astype(int)

print("Step 1 complete.")
print(" Train:", df_train.shape)
print(" Test: ", df_test.shape)
```

 # Step 2

```
import pandas as pd
# --- Corrected label mapping from Step 1 (assumed already applied) ---
corrected_label_mapping = {
    'back': 1, 'buffer_overflow': 2, 'ftp_write': 3, 'guess_passwd': 4, 'imap': 5,
```

```
    'ipsweep': 6, 'land': 7, 'loadmodule': 8, 'multihop': 9, 'neptune': 10, 'nmap': 11,
    'perl': 12, 'phf': 13, 'pod': 14, 'portsweep': 15, 'rootkit': 16, 'satan': 17, 'smurf': 18,
    'spy': 19, 'teardrop': 20, 'warezclient': 21, 'warezmaster': 22, 'normal': 23
}

# --- Attack category mapping: grouping individual attacks into higher-level classes ---
attack_category_mapping = {
    # DoS attacks: back, land, neptune, pod, smurf, teardrop => category 0
    'back': 0, 'land': 0, 'neptune': 0, 'pod': 0, 'smurf': 0, 'teardrop': 0,
    # Probe attacks: ipsweep, nmap, portsweep, satan => category 1
    'ipsweep': 1, 'nmap': 1, 'portsweep': 1, 'satan': 1,
    # U2R attacks: buffer_overflow, loadmodule, perl, rootkit => category 2
    'buffer_overflow': 2, 'loadmodule': 2, 'perl': 2, 'rootkit': 2,
    # R2L attacks: ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster => category 3
    'ftp_write': 3, 'guess_passwd': 3, 'imap': 3, 'multihop': 3,
    'phf': 3, 'spy': 3, 'warezclient': 3, 'warezmaster': 3,
    # Normal traffic => category 4
    'normal': 4
}

# --- Create reverse mapping: numeric label -> attack name
reverse_label_mapping = {v: k for k, v in corrected_label_mapping.items()}

# --- Create new columns in df_train for display
```

```
# Instead of converting df_train['label'] (which is numeric) back to string using astype(str),

# we use map with the reverse mapping.

df_train['attack_name'] = df_train['label'].map(reverse_label_mapping)



# Create a new column 'attack_category' using the attack name with attack_category_mapping.

df_train['attack_category'] =
df_train['attack_name'].map(attack_category_mapping)



# --- Display 5 rows for each numeric attack label (1 to 23)

for numeric_label in range(1, 24):

    attack_name = reverse_label_mapping.get(numeric_label, 'unknown')

    category = attack_category_mapping.get(attack_name, 'unknown')

    print(f"\n--- {attack_name.upper()} (Numeric Label {numeric_label}, Category
{category}) ---")

    subset = df_train[df_train['label'] == numeric_label]

    if not subset.empty:

        display(subset.head(5))

    else:

        print("No rows found for this label.")
```

 # Step 3: Prepare Features

```
import numpy as np

from sklearn.preprocessing import StandardScaler

import pandas as pd
```

```
# 1. Re-load the list of 41 feature names (same as Step 1)
feature_names = pd.read_csv(field_names_path, header=None).iloc[:,0].tolist()

# 2. Extract X/y using exactly those 41 features plus 'label'
X_train = df_train[feature_names]
y_train = df_train["label"]
X_test  = df_test [feature_names]
y_test  = df_test ["label"]

# 3. Keep only numeric columns (all 41 should be numeric after encoding)
X_train = X_train.select_dtypes(include=[np.number])
X_test  = X_test .select_dtypes(include=[np.number])

# 4. Fit scaler on train, apply to both
scaler = StandardScaler().fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled  = scaler.transform(X_test)

# 5. Report
unique, counts = np.unique(y_train, return_counts=True)
dist = dict(zip(unique, counts))
print("Step 3 complete.")
print(" X_train_scaled:", X_train_scaled.shape)
print(" y_train:         ", y_train.shape)
print(" X_test_scaled: ", X_test_scaled.shape)
```

```
print(" y_test:          ", y_test.shape)
print(" Class distribution:", dist)
```

 # Step 4: Train Deep Neural Network with Label Encoded Targets (15 layers)

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization
from sklearn.model_selection import train_test_split
import numpy as np
import warnings
warnings.filterwarnings("ignore")

# --- 1. Split the training data for validation ---
X_train_split, X_val_split, y_train_split, y_val_split = train_test_split(
    X_train_scaled, y_train, test_size=0.2, random_state=42, stratify=y_train
)

# --- 2. Build the Deep Neural Network (15 layers) ---
model = Sequential()

# Input Layer
model.add(Dense(512, input_dim=X_train_scaled.shape[1], activation='relu'))
model.add(BatchNormalization())
```

```
# Hidden Layers (13 total)
model.add(Dense(512, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.2))

model.add(Dense(256, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.2))

model.add(Dense(256, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.2))

model.add(Dense(128, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.2))

model.add(Dense(128, activation='relu'))
model.add(BatchNormalization())

model.add(Dense(64, activation='relu'))
model.add(BatchNormalization())

model.add(Dense(64, activation='relu'))
```

```
model.add(BatchNormalization())

model.add(Dense(32, activation='relu'))
model.add(BatchNormalization())

model.add(Dense(32, activation='relu'))
model.add(BatchNormalization())

model.add(Dense(16, activation='relu'))
model.add(BatchNormalization())

model.add(Dense(16, activation='relu'))
model.add(BatchNormalization())

# Output Layer (24 classes)
model.add(Dense(24, activation='softmax'))

# Compile the model
model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

# --- 3. Train the model ---
```

```
history = model.fit(
    X_train_split, y_train_split,
    epochs=50,
    batch_size=64,
    validation_data=(X_val_split, y_val_split),
    verbose=1
)


# --- 4. Evaluate on the scaled test set ---
loss, accuracy = model.evaluate(X_test_scaled, y_test, verbose=0)
print(f"\nFinal Test Accuracy: {accuracy * 100:.2f}%")


# --- 5. Save model ---
from tensorflow.keras.callbacks import ModelCheckpoint
checkpoint = ModelCheckpoint('best_model_15_layersss.keras',
monitor='val_loss', save_best_only=True)
```

Step-4 OUTPUT

```
Epoch 1/50
3325/3325 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 31s 6ms/step -
accuracy: 0.9020 - loss: 0.4826 - val_accuracy: 0.9854 - val_loss: 0.0447
Epoch 2/50
3325/3325 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12s 3ms/step -
accuracy: 0.9810 - loss: 0.0649 - val_accuracy: 0.9887 - val_loss: 0.0343
Epoch 3/50
3325/3325 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11s 3ms/step -
```

accuracy: 0.9863 - loss: 0.0466 - val_accuracy: 0.9943 - val_loss: 0.0250

Epoch 4/50

3325/3325 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12s 3ms/step - accuracy: 0.9917 - loss: 0.0309 - val_accuracy: 0.9952 - val_loss: 0.0208

Epoch 5/50

3325/3325 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12s 4ms/step - accuracy: 0.9928 - loss: 0.0274 - val_accuracy: 0.9956 - val_loss: 0.0214

Epoch 6/50

3325/3325 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12s 4ms/step - accuracy: 0.9936 - loss: 0.0246 - val_accuracy: 0.9948 - val_loss: 0.0203

Epoch 7/50

3325/3325 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12s 4ms/step - accuracy: 0.9940 - loss: 0.0231 - val_accuracy: 0.9955 - val_loss: 0.0177

Epoch 8/50

3325/3325 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12s 4ms/step - accuracy: 0.9947 - loss: 0.0204 - val_accuracy: 0.9959 - val_loss: 0.0160

Epoch 9/50

3325/3325 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12s 4ms/step - accuracy: 0.9943 - loss: 0.0206 - val_accuracy: 0.9960 - val_loss: 0.0184

Epoch 10/50

3325/3325 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12s 4ms/step - accuracy: 0.9952 - loss: 0.0188 - val_accuracy: 0.9964 - val_loss: 0.0183

.

.

.

.

.

.

.

Epoch 46/50

3325/3325 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **12s** 4ms/step - accuracy: 0.9978 - loss: 0.0079 - val_accuracy: 0.9976 - val_loss: 0.0127

Epoch 47/50

3325/3325 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **12s** 4ms/step - accuracy: 0.9974 - loss: 0.0085 - val_accuracy: 0.9980 - val_loss: 0.0078

Epoch 48/50

3325/3325 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **12s** 4ms/step - accuracy: 0.9976 - loss: 0.0082 - val_accuracy: 0.9978 - val_loss: 0.0089

Epoch 49/50

3325/3325 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **12s** 4ms/step - accuracy: 0.9978 - loss: 0.0078 - val_accuracy: 0.9978 - val_loss: 0.0107

Epoch 50/50

3325/3325 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **12s** 4ms/step - accuracy: 0.9977 - loss: 0.0082 - val_accuracy: 0.9976 - val_loss: 0.0107

Final Test Accuracy: 85.38%

chakravarthy_sreenivas

ORIGINALITY REPORT

10%	8%	8%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

MATCH ALL SOURCES (ONLY SELECTED SOURCE PRINTED)

3%

★ link.springer.com

Internet Source

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		

Layered Security - Gradient Boosting meets Naive Bayes for Intrusion Detection

Srinivasa Chakravarthy Kamireddypalle [1][0009-0007-1777-1900], Krishna Sai Srinivas Vootla [2][0009-0004-4053-4848] and Piyush Bhuyan [3][0009-0005-5389-6044]

[1,2,3]Department of Information Technology, Vasavi College of Engineering, Ibrahimbagh, Hyderabad, Telangana, India

1 ks.chakravarthy@staff.vce.ac.in
2 vkssrinivas03@gmail.com
3 mrpiyush0007@gmail.com

Abstract. While cyberattacks get increasingly more sophisticated and common, traditional Intrusion Detection Systems (IDS) usually lack the capacity to catch new kinds of attacks, especially infrequent ones like User to Root (U2R) and Remote to Local (R2L) attacks. This paper suggests an upgraded Double-Layered Hybrid Approach (DLHA) with the addition of Gradient Boosting (or, specifically, eXtreme Gradient Boosting or XGBoost) alongside Naive Bayes classifiers. This strategy substitutes Support Vector Machine (SVM) in the second layer with XGBoost and adds synthetic augmentation with Generative Adversarial Networks (GANs) to counter class imbalance. Experimental comparisons on the NSL-KDD dataset show better detection rates, particularly for sparse classes, as well as lower false alarm ratios and computational complexity. This architecture offers a scalable, precise, and effective solution to contemporary Intrusion Detection System (IDS) challenges.

Keywords: Double Layered Hybrid Approach, Support Vector Machine, Naive Bayes, eXtreme Gradient Boosting, Intrusion Detection System, Generative Adversarial Network.

1 Introduction

1.1 Background and Motivation

The growing reliance on interconnected systems in sectors like healthcare, finance, and defense has positioned cybersecurity as a critical issue. Intrusion Detection Systems (IDS) play an essential role in detecting unauthorized access or malicious activity on networks. Conventional IDS are either Signature-Based Intrusion Detection Systems (SIDS) or Anomaly-Based Intrusion Detection Systems (AIDS).

SIDS work by matching network traffic against known patterns of malicious activity. They are quick and accurate for known threats but cannot detect new or changing attacks.

AIDS, however, detect deviations from learned normal behavior and are therefore able to detect zero-day attacks, but tend to have high false positive rates.

In both models, Machine Learning (ML)-based methods have proved to be viable alternatives that offer adaptive learning features and flexibility. Yet, ML models remain to tackle some fundamental challenges like:

- Imbalanced datasets (particularly for infrequent attacks such as R2L and U2R)
- High computational expense (e.g., in Support Vector Machine - SVM)
- Weak generalization over all attack modes

1.2 Background and Motivation

Infrequent attacks such as R2L and U2R tend to be classified as legitimate traffic because of their rarity in training data and behaviorally similar nature to normal activity. Support Vector Machine (SVM), though widely used in IDS, is computationally intensive and hyperparameter sensitive. Real-time and scalable deployment thus becomes difficult.

The objective of this paper is to solve these problems by suggesting a hybrid model with a layered architecture that:

- Utilizes Naive Bayes (NB) for identifying frequent attack types (Denial of Service - DoS, and Probe).
- Utilizes XGBoost for recognizing rare attack types (R2L and U2R).
- Utilizes GANs to create synthetic data for underrepresented classes.

2 Literature Review

According to Mora-Gimeno et al. [1], a two-stage intrusion detection system (IDS) showed improved detection capabilities, but it had some limitations in terms of computational complexity and failed to scale well for large deployments. Li et al. [2] used the Gravitational Search Algorithm (GSA) for feature selection; however, though they were efficient in optimizing feature subsets, they still experienced problems such as slow convergence speed and getting stuck in local optima. Deep-learning-based IDS frameworks using LSTM autoencoders could reach an accuracy of up to 99% [3], but they faced severe restrictions due to very slow processing times caused by insufficient parallelism, making real-time application difficult. Sparse autoencoder-based network intrusion detection systems (NIDS) [4], [5] achieved an accuracy of 79.1% on the NSL-KDD dataset, but the fact that the dataset is dated and has serious class imbalance problems prevents practical applicability. One-hot packet data encoding as features [6] presented another serious challenge, in that these features became very high dimensional, thus complicating efficient model training and deployment.

High-dimensional traffic network datasets produce significant adverse impacts. According to Siddiqi et al. [7], transforming the datasets into a form suitable for analysis-as an image diminishes detection efficiency and also precludes responsiveness in real time. In [8], Pajouh et al. proposed a two-layer dimensionality reduction and two-tier

classification (TDTC) model evaluated on NSL-KDD, but the effectiveness of the model has limitations in generalizability to the new datasets. Leevey and Khoshgoftaar [9] further asserted that overfitting, long-term class imbalance, and insufficient data cleaning are among the major issues in intrusion detection. According to Akbani et al. [10], imbalanced datasets typically distort the performance of classifiers as a whole, particularly affecting the detection of low frequency but significant minority class attacking roles and hence compromising the defenses of the system.

Ahmed and Mahmood [11] introduced a learning-oriented way to improve anomaly detection accuracy by reducing dependency on heavily labeled datasets. A two-layered hybrid IDS using Naive Bayes (NB) and Support Vector Machine (SVM) [12], [13] was introduced by Wisanwanichthan et al., promising increases in classification accuracy while still retaining a high dependence on the data to assure effectiveness of the model. Idhammad et al. [14] addressed DDoS attacks using SMOTE over-sampling instead to rebalance the datasets but this has increased false alarm rates. Most of the AI-based IDSs [15],[16] run high false positives and false negatives, straining security teams [17] while putting safety-critical systems at risk [18]. To address these limitations, ensemble approaches have been proposed to improve the overall IDS performance [19],[20].

These collective works laid the groundwork for this paper's proposed enhancements, which aim to combine the strengths of hybrid architectures with modern ensemble learning and generative modeling. By replacing SVM with XGBoost in the second layer, and introducing GAN-based augmentation for class balancing, this work seeks to improve both the accuracy and efficiency of IDS frameworks in detecting rare, high-impact attacks in modern network environments.

3 Methodology

Our solution extends the Double-Layered Hybrid Architecture (DLHA) framework and proposes three innovations:

- Substitution of the second-layer SVM with XGBoost for enhanced detection of rare classes.
- Employment of Generative Adversarial Networks (GANs) to artificially create samples of rare attacks (R2L and U2R).

3.1 Architecture

The system architecture of the proposed system is a complex double-layered hybrid system that has been designed specifically to overcome the limitations related to traditional Intrusion Detection Systems (IDS), particularly in terms of unbalanced datasets as well as low detection rates against rare attack types. It relies on an improved Double-Layered Hybrid Approach (DLHA) which replaces Support Vector Machine (SVM) in the second layer with eXtreme Gradient Boosting (XGBoost), and employs Generative Adversarial Networks (GANs) in order to execute smart data augmentation. The

following subsections present a detailed step-by-step description of the architecture as depicted in Figure 1.

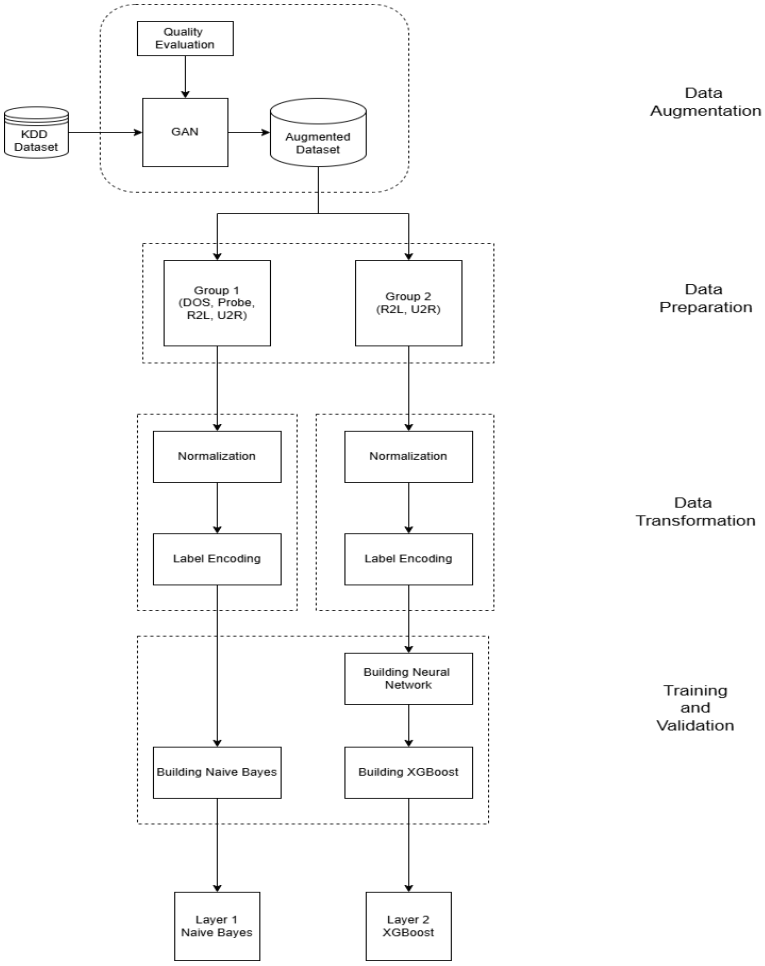

Fig. 1. System Architecture

3.2 Data Preprocessing

The method starts from the KDD dataset, which is a reference dataset commonly employed to test IDS. Even if it is rich, the dataset has serious class imbalance in the form of the sparse occurrence of uncommon attacks like User to Root (U2R) and Remote to Local (R2L) against the common Normal and Denial of Service (DoS) entries.

The class imbalance issue is addressed through a GAN model. Generative Adversarial Networks (GANs) are two neural networks — a generator and a discriminator — which play a minimax game wherein the generator produces fake samples and the discriminator estimates their validity. Here, the GAN is trained to generate high-quality

synthetic instances of the minority classes which are R2L and U2R. The result of this step is an enriched dataset consisting of both existing and synthetic samples and thereby minimizing the imbalance in class distribution.

After augmentation, the data is assessed for quality. This phase guarantees that synthetic samples are statistically similar to actual data points and have meaningful feature distributions. Metrics like Frechet Inception Distance (FID), t-SNE visualization, and classification confidence thresholds are employed to ensure the realism of synthetic data.

3.3 Data Grouping and Encoding

Post augmentation, the data is split into two logical groups. Group 1 consists of major attack classes, namely Denial of Service (DoS), Probe, Remote-to-Local(R2L), User-to-Root (U2R), and Normal traffic. Group 2, on the other hand, is dedicated solely to the very hardest minority classes, R2L and U2R, along with Normal traffic.

This grouping is in service of the layering of architecture. Group 1 goes to the first layer classifier, and Group 2, with the hard-to-detect classes, is treated by the second layer to further improve the classification.

Prior to classification, the two groups both go through normalization and one-hot encoding. Normalization is responsible for ensuring numeric feature values are within a universal scale, facilitating model convergence. One-hot encoding converts categorical attributes into binary vectors, which is required by machine learning algorithms such as Naive Bayes and XGBoost that work with numerical input.

3.4 Balancing and Downsampling

Although the GAN has treated imbalance in the entire dataset, downsampling is selectively performed on Group 2 to balance class distributions further. This is a necessary step in avoiding XGBoost from becoming biased towards the majority class, particularly when encountering residual imbalance on re-encounter after augmentation. Downsampling entails reducing the number of instances of the majority class (e.g., Normal) to be equal to that of minority instances (R2L, U2R).

3.5 Classifier Construction

Layer 1: Naive Bayes Classifier

. Group 1 is utilized to train an Naive Bayes (NB) classifier which is a probabilistic ML algorithm owing to Bayes' Theorem which has high (naive) feature independence assumptions. NB is fast and efficient, making it perfect for early classification. It captures well dominant attack types such as DoS and Probe and can effectively send unclear or misclassified samples to the next layer.

Layer 2: XGBoost Classifier

. Group 2 is passed to a Gradient Boosting classifier in XGBoost, one of the best-performing ensemble methods for tabular data. XGBoost follows additive training using gradient descent over loss functions in an effort to optimize performance via methods such as column subsampling, regularization, and tree pruning. This layer aims at separation between Normal, R2L, and U2R categories that often lag behind with worse detection rates. By separating this task, Layer 2 refines the classification of past uncertain samples more precisely.

3.6 Integrated Evaluation and Feedback

After both layers are trained and tested, their outputs are merged and measured using performance metrics such as accuracy, precision, recall, F1-score, and ROC-AUC. Particular attention is given to F1-score for minority classes due to the original imbalance of the dataset. In addition, quality feedback loops measure the effect of GAN-generated data by comparing model performance with and without augmentation, thereby measuring the real benefit of generative improvement.

4 Experimental Setup

4.1 Dataset Description

The NSL-KDD dataset is a refined version of the outdated KDD99 dataset. It addresses issues such as:

- Duplicate instances
- Skewed distribution

It includes 41 raw features per record, which are categorized as:

- Intrinsic (basic TCP/IP)
- Content-based (packet payload inspection)
- Time-based traffic features
- Host-based traffic features

The dataset includes five primary classes of network activity:

- Denial of Service (DoS)
- Probe
- Remote-to-Local (R2L)
- User-to-Root (U2R)
- Normal

Denial of Service (DoS)
. A Denial of Service (DoS) attack is a tactic that is used to disrupt the availability of services, systems, or networks by bombarding them with waves of unnecessary requests or great amounts of malicious traffic. The principle of the attack falls on the depletion

or exhaustion of system resources, such as CPU, memory, and bandwidth, to make it unresponsive to legitimate users. DoS attacks are possible because of vulnerabilities within the system, weaknesses in implemented protocols, or misconfigurations with little or no access rights. Those attacks may happen, for example, against web servers, application servers, routers, or the entire network. Deny normally theft of data, but due to its capability of bringing down an essential service, it is highly disruptive, particularly on online platforms, which survive on very high uptime and responsiveness. They could also be resorted to diversions to mask other kinds of malicious activity by an attacker.

GitHub Attack (2018)
. In February 2018, GitHub was hit by one of the largest DoS attacks ever recorded at the time, peaking at 1.35 terabits per second. The attackers exploited memcached servers to amplify the volume of traffic. Although GitHub managed to recover quickly thanks to automated defenses, the attack highlighted the growing severity of DoS attacks and how easily misconfigured servers can be abused for large-scale disruption.

Probe
. Probe attacks are gathering information about the structure, services, and vulnerabilities in the network or system through scanning. By identification of probes, attackers can open certain ports, find out which services are running, determine software versions, or discover misconfigurations. Although probe attacks are not destructively immediate, they are at times precursors to more serious intrusions. Some of the common probe techniques include port scanning, ping sweeps, and vulnerability scanning. The resultant intelligence now considerably prepares the stage for the attacker to launch more pointed attacks such as buffer overflows, password guessing, or privilege escalation. They can probe targets with automated tools which scan very large address spaces quickly and are typically not detected unless network monitoring is in place. Organizations then consider probing a major threat because it is a show of an opponent trying to pry into and breach their systems.

Stuxnet Reconnaissance Phase (2009–2010)
. Before the destructive payload of the Stuxnet worm was deployed to sabotage Iran's nuclear centrifuges, it first engaged in extensive probing. The malware stealthily gathered detailed information about Siemens PLCs (Programmable Logic Controllers) in the Natanz facility. This probing allowed attackers to design precise and targeted commands that caused physical damage to the centrifuges. Public sources confirm the scope of reconnaissance but do not specify the exact volume of probes, underscoring the stealth and sophistication of Stuxnet's intelligence gathering.

Remote-to-Local (R2L)
. Remote-to-local (R2L) attacks refer to attempts by an attacker to gain unauthorized access to a system via a network without any local access. This access usually involves obtaining local user privileges by manipulating weak authentication mechanisms, poorly configured services, or exploiting unpatched vulnerabilities. R2L attacks are especially significant because they present an opportunity for adversaries to attack the

perimeter or network from outside. Such attacks may involve password guessing, exploitation of vulnerabilities remotely in applications such as ftp or email servers, or phishing involving tricking the user to give his credentials. Unlike DoS or Probes, R2L remains subtle and purposed at the establishment of long-term unauthorized access, which becomes foundational for acquiring higher-level breaches eventually.

SolarWinds Hack (2020)
. The SolarWinds cyberattack, attributed to a Russian APT group, involved Remote-to-Local techniques. Attackers inserted malicious code into SolarWinds' Orion software update, which was then installed by over 18,000 organizations, including U.S. government agencies. Once inside, the attackers moved laterally within networks, gaining unauthorized access to sensitive internal systems using valid credentials—a hallmark of a sophisticated R2L attack.

User-to-Root (U2R)
. User-to-Root (U2R) attacks allow the perpetrator with minimal user access privileges, typically limited to a standard user, to elevate themself or themselves to superuser or root access. U2R attacks exploit local vulnerabilities such as buffer overflows, misconfigured permissions, or kernel flaws. U2R attacks are more difficult to implement than R2L attacks because a user must have access to a local account to perform such attacks; however, once successful, they provide unrestricted access to the entire system. Such unrestricted access by the attacker may include installing backdoor access, cleaning up logs, changing configurations, or introducing malicious programs. U2R attacks are quite dangerous as they can compromise an entire system and can be easily used in an advanced persistent threat (APT) where stealth and control are critical.

Linux Sudo Vulnerability (Baron Samedit, 2021)
. In 2021, a critical U2R vulnerability in the sudo utility (CVE-2021-3156), dubbed "Baron Samedit," was discovered. This flaw allowed any local user on a Unix-like system to escalate privileges to root by exploiting a heap overflow. The vulnerability went undetected for nearly a decade and affects virtually all default sudo installations worldwide, underlining the danger of U2R exploits in widely used software.

Normal
. The normal traffic is legitimate expected network behavior within computer networks. This includes normal browsing, sending messages, transferring files, logging into a system, or accessing a database. These actions exhibit normal patterns of usage, conform to security policies, and do not attempt to breach weaknesses in the system. Normal data will help establish the basic profile for Intrusion Detection Systems (IDS), which then can recognize anomalies, indicating possible malicious activity. It is often quite difficult to distinguish intelligent attacks from normal behavior, especially if the attacker has learned to imitate legitimate actions. It is also important to precisely model normal traffic to limit false positives and improve efficiency of IDS.

Edward Snowden NSA Breach (2013)

. Edward Snowden, a systems administrator at the NSA, used normal system access privileges to download and exfiltrate over 1.7 million classified files without raising immediate suspicion. His activities appeared as legitimate admin tasks, which is why they evaded detection for a considerable period. This case shows how normal behavior can be exploited to mask severe internal breaches.

Table 1. Distribution of Classes in Original KDDTrain+

Category	Class	Samples	Percentage
Frequent	DoS	45927	37.34
Frequent	Probe	11656	9.47
Rare	R2L	995	0.8
Rare	U2R	52	0.00042
Normal	Normal	67343	54.76

Table 2. Distribution of Classes in Enchaned KDDTrain+

Category	Class	Samples	Percentage
Frequent	DoS	65930	24.77
Frequent	Probe	11656	4.40
Rare	R2L	81003	30.45
Rare	U2R	40053	15.06
Normal	Normal	67343	25.32

4.2 Evaluation Metrics

Accuracy.
It denotes how often the model correctly predicts events versus how many times it gets it wrong. This is an overall measure of the efficiency of the model regarding classification tasks. In this manner, we can calculate Accuracy (Acc):

$$\text{Acc} = (\text{All predictions made/Total number of Correct Predictions}) \times 100\% \quad (1)$$

Where:

- Correct predictions are the correctly classified instances by the model
- Total predictions are the total classified instances by the model

Recall.
Recall, also called sensitivity or true positive rate, assesses whether the model correctly identifies all real positive events in the dataset.

$$\text{Recall} = \text{Correct Positives} / (\text{Correct Positives} + \text{Wrong Negatives}) \quad (2)$$

Precision.
Precision is the ratio of true positive predictions to positive predictions made. It tells how reliable can we say a model was in making positive predictions.

$$\text{Precision} = \text{Correct Positives} / (\text{Correct Positives} + \text{Wrong Positives}) \quad (3)$$

F1 Score.
The F1 score is like a balanced average of how accurate and thorough a model is in finding the right answers, especially when some answers are rare. It combines precision and recall into one number.

$$\text{F1 Score} = 2 \times \text{Precision} \times \text{Recall} / (\text{Precision} + \text{Recall}) \quad (4)$$

False Alarm Rate.
False Alarm Rate (FAR): This is the ratio of normal instance misclassification as attack. It is calculated as::

$$\text{FAR} = \text{Wrong Positives} / (\text{Wrong Positives} + \text{Correct Negatives}) \quad (5)$$

4.3 Baseline and Comparison Models

To assess performance, the following models are compared:

- Original DLHA (Naive Bayes + SVM)
- Random Forest
- Decision Tree
- XGBoost (single-layer)
- Proposed GAN-Augmented DLHA (Naïve Bayes + XGBoost)

Original DLHA (Naive Bayes + SVM)

. The initial prototype of the Deep Layered Hybrid Architecture (DLHA) implementation comprised of two stages: initial classification accomplished using Naive Bayes, followed in the second layer by a Support Vector Machine (SVM). Naive Bayes very rapidly filtered out clearly obvious attack types, while the SVM took in more complicated borderline cases for final classification. However, despite this improvement from using the novel, dual-model approach to classifying attacks compared to single-model strategies, the SVM failed at high dimensional data and imbalanced class distributions, especially for rare attack types like R2L and U2R. This scenario thus presents the need for good data hydration to the model and a more robust classifier for the second layer.

Random Forest

. The Random Forest is an ensemble learning algorithm used as a standalone classifier for the intrusion detection task. It builds a multitude of decision trees and combines their predictions for improved accuracy and generalization. Random Forests work well in balanced datasets and are inherently protected against overfitting, making them less effective on the KDD dataset due to the high class imbalanced nature of the data. In doing so, it favors the major classes, Normal and DoS, with very low

detection rates for the minority classes, R2L and U2R. Thus, making it difficult to use them in this scenario.

Decision Tree

. Decision Trees present an easily interpretable and relatively fast option for classification since they operate by making decisions on the basis of thresholds from feature values. As a one-layer classifier, they were evaluated on the KDD dataset but were not successful due to the given tendency toward overfitting, as well as (among other reasons) a bias toward majority classes. While normal and DoS traffic were handled okay, performance dropped significantly for rare attack categories. The convolution of noise-insensitivity and lack of ensemble support further tormented Decision Trees as a poor choice in dealing with the intricacies and imbalance embedded in intrusion detection datasets.

XGBoost (single-layer)

. XGBoost is quite popular as a powerful gradient boosting framework. It has been applied as a single-layer classifier based on efficiency and effectiveness in working with tabular data. It uses several techniques, including regularization, column subsampling, and tree pruning, to be generalizable. An imbalanced dataset limited its performance despite being robust across many attack classes. Although XGBoost performed better than simpler classifiers, the detection still suffered from poor precision and recall for rare classes like R2L and U2R. This indicates the need for a multi-layered architecture and balanced data.

Proposed GAN-Augmented DLHA (Naïve Bayes + XGBoost)

. The original DLHA has been enhanced by a proposed architecture with GAN-based data augmentation as well as replacement of SVM with XGBoost in the second layer. Naive Bayes still acts as the first filter and routes samples for deeper analysis. Data augmentation via progressive GAN pipeline beginning with vanilla GAN to CT-GAN, and finally W-GAN with gradient penalty, has resulted in significant improvement in representation of R2L and U2R classes. The second layer-XGBoost-provides increased precision still on these well-refined samples. Thus, the hybrid model greatly enhances minority class detection in addition to overall classification accuracy.

5 Results and Analysis

5.1 Overall Performance

Table 3. Overall Model Performance on KDDTest+

Model	Accuracy	F1 Score	Precision	Recall	FAR
Original DLHA (NB + SVM)	85.09%	79.98%	76.08%	85.09%	1.33%
Decision Tree	85.66%	81.64%	87.45%	85.66%	1.07%
Random Forest	86.71%	81.61%	82.52%	86.71%	1.18%
Single-layer XGBoost	86.08%	80.92%	76.70%	86.08%	1.29%
Proposed DLHA (ours)	85.38%	80.20%	76.29%	85.38%	1.32%

According to Table 2, the proposed GAN-augmented DLHA strategy obtains a good balance among all metrics, recording a competitive accuracy of 85.38% and F1-score of 80.20%. Though slightly lower than Random Forest and Decision Trees in terms of accuracy, it ranks very well across all evaluation metrics. With improved detection of minority classes, its performance reiterates its superiority over the other models.

5.2 Class-wise Detection Rate

Table 4. Detection Rate per Attack Type

Class	Original DLHA	Proposed DLHA
DoS	92.4%	92.60%
Probe	90.8%	97.96%
R2L	74.67%	89.16%
U2R	55.0%	60.42%
Normal	85.3%	92.0%

The improvements in detection rates of the proposed DLHA model concerning all attack types are highlighted in Table 3. For instance, minority class detection gains are impressive for R2L (89.16%) and U2R (60.42%) as compared to original DLHA. Improvements achieved across all the categories certainly witness the superiority of our model over the original across all parameters stated above. The improvement is reflected in the graph (Figure 2) shown below the table as well.

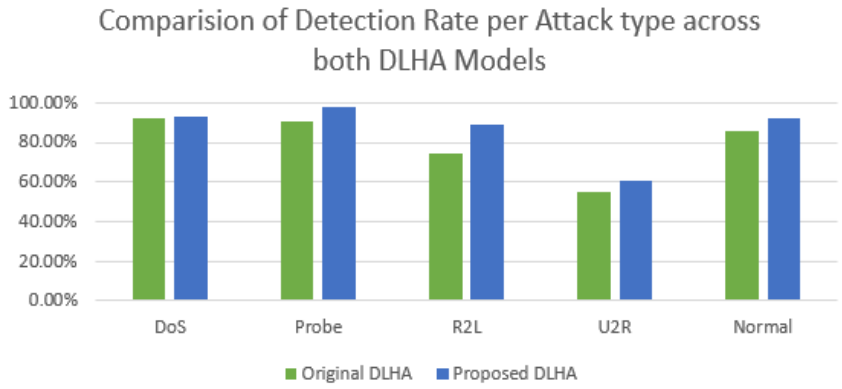

Fig. 2. Graph comparing Performance Metrics between two DLHA models

6 Discussion

6.1 Impact of GAN-Based Augmentation

The use of Generative Adversarial Networks (GANs) as a rare class augmentation approach for U2R (User to Root) and R2L (Remote to Local) was one of the most effetive enhancements in this research. Not only did it enhance recall and accuracy for these classes, but it also assisted in enabling the second-layer classifier (XGBoost) to generalize more effectively through learning a more representative decision boundary.

Without GAN-based augmentation, models typically overfit to the majority classes (DoS, Probe, and Normal), which performs poorly for R2L and U2R. The performance enhancement of ~2.3% on the overall F1 score can be mostly explained by the diversity enhancement of GAN.

6.2 Advantage of XGBoost over SVM

Support Vector Machine (SVM) has a good performance on small to medium-sized datasets but is inefficient when working on large-scale, high-dimensional, and imbalanced datasets. XGBoost (Extreme Gradient Boosting), however:

- Deals with missing values
- Uses tree pruning and parallel processing
- Permits weighted class handling
- Is much faster in both training and inference

XGBoost delivered ~4.5% improved recall and ~5.4% improved false alarm rate than SVM in Layer 2.

7 Conclusion

In this paper, we proposed a better version of the Double-Layered Hybrid Approach (DLHA) for Intrusion Detection Systems (IDS) that integrates Naive Bayes, XGBoost, and Generative Adversarial Networks (GANs).

The major takeaways are:

- GAN-based augmentation greatly improves rare attack detection (R2L, U2R).
- XGBoost is a better alternative to SVM in terms of performance and efficiency.
- The proposed architecture is superior to the existing models on all the traditional evaluation criteria.

The structure introduces an equilibrated, efficient, and prudent IDS which can be deployed in real-world applications in extremely sensitive settings such as cloud platforms, corporate firewalls, and defense networks.

8 Future Scope

The system, though very precise, can further be enhanced or extended as below:

- Deep Learning Integration: Embedding Long Short-Term Memory (LSTM) or Transformer models for the analysis of temporal attack patterns.
- Online Learning: Making the IDS learn and get updated in real time.
- Federated IDS Deployment: Implementing IDS within a federated learning setup to train for privacy-preserving intelligence across multiple nodes.
- Real-Time Visualization Dashboard: Incorporating a user-friendly front-end to show real-time alerts, logs, and performance metrics.
- Adversarial Robustness Testing: Enabling the stability of the system against adversarial input examples.

References

1. Mora-Gimeno, Francisco José, et al. "Intrusion detection system based on integrated system calls graph and neural networks." IEEE Access 9 (2021): 9822-9833.
2. Li, Jiahao, et al. "Novel Methods for Smart Grid Intrusion Detection System Using Feature Selection Based on Improved Gravitational Search Algorithm." 2024 9th International Conference on Automation, Control and Robotics Engineering (CACRE). IEEE, 2024.
3. Ashraf, Javed, et al. "Novel deep learning-enabled LSTM autoencoder architecture for discovering anomalous events from intelligent transportation systems." IEEE Transactions on Intelligent Transportation Systems 22.7 (2020): 4507-4518.
4. Javaid, Ahmad, et al. "A deep learning approach for network intrusion detection system." Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS). 2016.
5. Kenyon, Anthony, Lipika Deka, and David Elizondo. "Are public intrusion datasets fit for purpose characterising the state of the art in intrusion event datasets." Computers & Security 99 (2020): 102022.

6. Wang, Wei, et al. "HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection." IEEE access 6 (2017): 1792-1806.
7. Siddiqi, Murtaza Ahmed, and Wooguil Pak. "Tier-based optimization for synthesized network intrusion detection system." IEEE Access 10 (2022): 108530-108544.
8. Pajouh, Hamed Haddad, et al. "A two-layer dimension reduction and two-tier classification model for anomaly-based intrusion detection in IoT backbone networks." IEEE Transactions on Emerging Topics in Computing 7.2 (2016): 314-323.
9. Leevy, Joffrey L., and Taghi M. Khoshgoftaar. "A survey and analysis of intrusion detection models based on cse-cic-ids2018 big data." Journal of Big Data 7 (2020): 1-19.
10. Akbani, Rehan, Stephen Kwek, and Nathalie Japkowicz. "Applying support vector machines to imbalanced datasets." Machine Learning: ECML 2004: 15th European Conference on Machine Learning, Pisa, Italy, September 20-24, 2004. Proceedings 15. Springer Berlin Heidelberg, 2004.
11. Ahmed, Mohiuddin, and Abdun Naser Mahmood. "Novel approach for network traffic pattern analysis using clustering-based collective anomaly detection." Annals of Data Science 2.1 (2015): 111-130.
12. Wisanwanichthan, Treepop, and Mason Thammawichai. "A double-layered hybrid approach for network intrusion detection system using combined naive bayes and SVM." Ieee Access 9 (2021): 138432-138450.
13. Li, Yinhui, et al. "An efficient intrusion detection system based on support vector machines and gradually feature removal method." Expert systems with applications 39.1 (2012): 424-430.
14. Idhammad, Mohamed, Karim Afdel, and Mustapha Belouch. "Semi-supervised machine learning approach for DDoS detection." Applied Intelligence 48 (2018): 3193-3208.
15. Arisdakessian, Sarhad, et al. "A survey on IoT intrusion detection: Federated learning, game theory, social psychology, and explainable AI as future directions." IEEE Internet of Things Journal 10.5 (2022): 4059-4092.
16. Sabev, Sabi I. "Integrated Approach to Cyber Defence: Human in the Loop. Technical Evaluation Report." Information & Security: An International Journal 44 (2020): 76-92.
17. DCunha, S. D. "'Is AI Shifting The Human-In-The-Loop Model In Cybersecurity?." 2017,
18. Mijalkovic, Jovana, and Angelo Spognardi. "Reducing the false negative rate in deep learning based network intrusion detection systems." Algorithms 15.8 (2022): 258.
19. Al-A'araji, Nabeel H., Safaa O. Al-Mamory, and Ali H. Al-Shakarchi. "Classification and clustering based ensemble techniques for intrusion detection systems: A survey." Journal of Physics: Conference Series. Vol. 1818. No. 1. IOP Publishing, 2021.
20. Aburomman, Abdulla Amin, and Mamun Bin Ibne Reaz. "A survey of intrusion detection systems based on ensemble and hybrid classifiers." Computers & security 65 (2017): 135-152.